Technical report, IDE1006, June 2010

Wireless Sensor Network Group Connectivity

Master's Thesis in Computer Network Engineering

Alia Ibrahim

&

Samar Sajadian

School of Information Science, Computer and Electrical Engineering
Halmstad University
Wireless Sensor Network Group Connectivity

Master Thesis in Computer Network Engineering

School of Information Science, Computer and Electrical Engineering
Halmstad University
Box 823, S-301 18 Halmstad, Sweden

June 2010
Preface

We are indebted to our parents; this thesis is dedicated to them for their years of love, support and help.

Our appreciation and thanks for the accomplishment of this study is directed to Prof. Tony Larsson for his guidance of this thesis.

Last, but not least, we thank our supervisor Edison Pignaton de Freitas for keeping us on track and giving us valuable advices.

Samar Sajadian & Alia Ibrahim
Halmstad University, June 2010
Wireless Sensor Network Group Connectivity
Abstract

The importance of monitoring physical and environmental conditions increases day by day and, therefore, so is the necessity of having a reliable wireless sensor network (WSN).

The need to overcome challenges in WSN deployment and operation arises due to WSN's nature and characteristics such as possible nodes' mobility, limited radio and processing power, available storage and physical effects of the environment (particularly harsh environments) and balancing energy consumption has motivated us to investigate solutions to those problems.

By studying related work, it was possible to observe that techniques such as the use of a good link estimator and the deployment of a suitable topology are essential features for a WSN. The core idea is to capture link connectivity dynamically and use it on routing decisions to gain reliability and estimate the whole network connectivity. The three main steps for deployment of a reliable WSN are the following:

- Link estimator
- Routing and neighbor information
- Suitable routing algorithms

In addition, self-organization is an important capability that WSNs need to present. They should be reliable, scalable and energy efficient during the network lifetime and self-organization plays a key role in this context.

Summing up all these aspects, it comes to the point that reliable connectivity is an important characteristic of a WSN. The goal of this work is to contribute with the research in the subject by means of implementing a suitable topology management and evaluating the network connectivity by the means of quantitative metric for the network as whole. Practical experiments results are presented and discussed.
## Contents

PREFACE .......................................................................................................................... 1

ABSTRACT ......................................................................................................................... 3

CONTENTS ......................................................................................................................... 5

LIST OF TABLES .................................................................................................................. 7

LIST OF FIGURERS ............................................................................................................. 8

LIST OF FORMULAS ............................................................................................................. 9

INTRODUCTION .................................................................................................................... 11

1.1 APPLICATION AREA AND MOTIVATION ................................................................ 11

1.2 PROBLEM STUDIED ............................................................................................... 12

1.3 APPROACH CHOSEN TO SOLVE THE PROBLEM ............................................... 12

1.4 THESIS GOALS AND EXPECTED RESULTS ........................................................ 12

2 BACKGROUND ............................................................................................................. 14

2.1 STATE OF THE ART .............................................................................................. 14

2.2 RELATED WORK .................................................................................................... 15

3 SOLUTION IDEA TO BE INVESTIGATED .................................................................... 17

3.1 METHODOLOGY ..................................................................................................... 17

3.2 LINK METRIC .......................................................................................................... 18

3.2.1 Chosen Link Metric for the experiments .......................................................... 19

3.3 WSN TOPOLOGIES ................................................................................................ 20

3.3.1 Star topology ..................................................................................................... 21

3.3.2 Mesh topology .................................................................................................. 21

3.3.3 Clustered hierarchical topology ........................................................................ 22

3.3.4 Clustering basis ................................................................................................ 23

3.3.5 Available clustering algorithms ........................................................................ 24

3.3.6 Chosen topology for the experiment ............................................................... 25

3.4 NETWORK CONNECTIVITY .................................................................................... 28

3.4.1 Graph theory ..................................................................................................... 29

3.4.2 Connectivity Formula ....................................................................................... 29

4 DETAILED DESCRIPTION OF THE INVESTIGATED SOLUTION ............................. 33

4.1 PHYSICAL OR HARDWARE STRUCTURE ................................................................ 33
Wireless Sensor Network Group Connectivity

4.2 LOGICAL OR SCHEMATIC STRUCTURE ......................................................................................... 35
  4.2.1 Algorithm description ................................................................................................................. 35
  4.2.2 How K value in the link metric formula is assigned for each node role ................................... 38
  4.2.3 Link metric usage ........................................................................................................................ 41
  4.2.4 Clusterhead selection .................................................................................................................... 41
  4.2.5 Clusters shaping (development, forming, arrangement, configuration) ...................................... 42
  4.2.6 How connectivity is calculated .................................................................................................... 42

4.3 BEHAVIOURAL OR FUNCTIONAL DESCRIPTION ........................................................................ 45
  4.3.1 Scenario 1: a node going far from the network ........................................................................ 45
  4.3.2 Scenario 2: a clusterhead going far from the network ............................................................... 47
  4.3.3 Scenario 3: a node/clusterhead dies .......................................................................................... 49
  4.3.4 Scenario 4: addition of a new node ............................................................................................ 49
  4.3.5 Scenario 5: minimum number of clusterheads are reached in the network ............................. 49

5 TEST OF SOLUTION ........................................................................................................................... 52

6 RESULTS OF TEST, MEASUREMENT OR SIMULATION .............................................................. 57

7 CONCLUSIONS AND SUGGESTIONS TO FUTURE WORK ........................................................... 59

8 APPENDIX ............................................................................................................................................. 65
Introduction

List of Tables

Table 1 - Nodes Specification ................................................................................................................................ 35
Table 2 - Timers ..................................................................................................................................................... 36
Table 3 - LEDs colors ............................................................................................................................................ 45
Table 4 - first test result ......................................................................................................................................... 53
Table 5 - Test result - a simple node moved far ..................................................................................................... 55
Table 6 - Test result - an empty clusterhead moved far ........................................................................................ 55
Table 7 - Test result - a clusterhead moved far .................................................................................................... 55
Table 8 - Test result - One node moved to two different distances ........................................................................ 56
List of Figures

Figure 1 - Star topology, Gray node: base station, Green node: simple node .............................................................. 21
Figure 2 - Tree topology, Dark gray node: base station, gray node: simple node, parent, Green node: simple node either parent or child, Blue node: simple node, child .............................................................................. 21
Figure 3 - Mesh topology ........................................................................................................................................... 22
Figure 4 - Clustered hierarchical ............................................................................................................................ 23
Figure 5 - Sunspot ..................................................................................................................................................... 33
Figure 6 - Sunspot...................................................................................................................................................... 34
Figure 7 - Network establishment flowchart ........................................................................................................ 37
Figure 8 - Network connectivity using set one for metric ...................................................................................... 38
Figure 9 - Network connectivity using set two for metric ..................................................................................... 38
Figure 10 - Network connectivity using set three for metric ................................................................................ 39
Figure 11 - Network connectivity using set four for metric .................................................................................. 39
Figure 12 - Network connectivity using set five for metric .................................................................................. 40
Figure 13 - Network connectivity comparison for different sets of K value in metric formula .................................. 40
Figure 14 - Network connectivity graph for 5 sets of metric ................................................................................ 41
Figure 15 - Connectivity procedure ..................................................................................................................... 44
Figure 16 - Scenario one procedure ..................................................................................................................... 46
Figure 17 - Scenario two ....................................................................................................................................... 48
Figure 18 - Scenario four .................................................................................................................................... 50
Figure 19 – Established network, proposed algorithm, first and second run ......................................................... 53
Figure 20 – Established network, random algorithm, first run .............................................................................. 54
Figure 21 – Established network, random algorithm, second run ........................................................................ 54
Figure 22 - Network comparison .......................................................................................................................... 57
Figure 23 - Connectivity comparison between proposed algorithm and random algorithm ................................ 57
Figure 24 - Effect of mobility on network connectivity ........................................................................................ 58
Figure 25 - Effect of two different distance of connectivity .................................................................................. 58
Introduction

List of Formulas

*Formula 1*- Link metric ................................................................................................................................. 20
*Formula 2*- Graph Formula ............................................................................................................................. 29
*Formula 3*- Basic reachability formula ......................................................................................................... 30
*Formula 4*- Connectivity formula .................................................................................................................. 30
*Formula 5*- Connectivity formula to measure link behavior effect ................................................................. 31
*Formula 6*- Cluster quality ............................................................................................................................. 31
*Formula 7*- Total network clusters quality .................................................................................................... 31
*Formula 8*- Power at radio frequency pins .................................................................................................... 34
*Formula 9*- Packet RSSI ................................................................................................................................. 41
*Formula 10*- Packet LQI ................................................................................................................................. 41
*Formula 11*- Clusterhead selection link metric .............................................................................................. 41
*Formula 12*- Joining clusters link metric ........................................................................................................ 42
Wireless Sensor Network Group Connectivity
Introduction

1.1 Application Area and Motivation

WSN is a network composed of sensor nodes (possibly mobile), which use their own power (generally supplied by a battery) to communicate with other nodes located at different distances, via radio frequency signals. The aim is to gather information from the nodes that are spread in an area, and send their information to a base station for further usage.

Wireless sensor networks are widely in use in various applications and areas lately. Therefore, supporting each environment's requirements is essential. This demand requires a concerted effort to understand the hardware and software characteristics of each application and a cooperation of experts to solve the problems, which arise due to applications' behavior. One solution can be to divide applications based on their needs by considering WSN characteristics like mobility, topology and deployment, energy, cost, network lifetime, and coverage area [1].

Some sample applications are: Great Duck (bird observation on Great Duck island), ZebraNet, Glacier (glacier monitoring), Herding (cattle herding), Bathymetry, Ocean (ocean water monitoring), Grape (grape monitoring), Cold Chain (cold chain management), Avalanche (rescue of avalanche victims), Vital Sign (vital sign monitoring), Power (power monitoring), Assembly (parts assembly), Tracking (tracking military vehicles), Mines (self-healing mine field) and sniper (sniper localization) [2]. Also there are several research groups working on different aspects of WSN like: SensorNet, WINS, SPINS, SINA, and mAMPS, LEACH, SmartDust, SCADDS, PicoRadio, PACMAN, Dynamic Sensor Networks, Aware Home COUGAR and Device Database Project DataSpace [3].

Different WSN applications can be used in different areas. An industrial WSN application is used for monitoring and controlling industrial equipment (LRWPAN [4]), controlling factory processes and industrial automation [5] and monitoring manufacturing [6]. In the military area, wireless sensor networks can be used for the sensing intruders on bases, the detection of enemy units' movements on land/sea, chemical/biological threats and offering logistics in urban warfare [7], battlefield surveillance [6] and military situation awareness [5]. WSN applications can be deployed on airports for smart badges and tags [8, 9] and wireless luggage tags [8]. In the agriculture field, WSN is used for sensing of soil moisture, pesticide, herbicide, and pH levels [8, 9]. Meanwhile, WSN can be used for public safety purposes like sensing and location determination at disaster sites [8, 9] or, alternatively, for emergencies like hazardous chemical levels and fires (petroleum sector) [8], fire/water detectors [10] and monitoring disaster areas [11].

In the broad range of WSN applications, the concept of connectivity is especially important when the nodes are mobile. It is important to understand how the nodes keep their connectivity with the others, in order to assess how reliable the communication in the network as a whole is.

Estimation of wireless sensor network connectivity is based on communication events and sharing the information gained by each node to achieve a successful measure.

The problems posed when using WSN are the following:

- Mobility nature of the nodes
- Limited power of the nodes
Wireless Sensor Network Group Connectivity

- Congestion and collision
- Poor and alternate channel conditions which is caused by environmental changes, obstacles in between, distance and loss rate (loss factors).
- Balancing energy consumption

1.2 Problem Studied

The mobility nature of the nodes poses problems in topology and energy management. Considering mobile nodes, the management of their connectivity is a major issue. The purpose of the thesis is to come up with a model that can minimize connectivity problems in WSN composed of mobile nodes, improve and estimate network connectivity. In order to search for a solution for WSN problems, nodes' pattern distribution, path reliability support and route stability in the network are considered.

Since most of the WSNs are deployed as static networks, a great number of researchers have put effort on studying fixed structure WSN problem. However, mobility is becoming an important aspect, which deserves attention [12]. Some approaches had been studied [40, 39] to be able to evaluate the connectivity in WSN [37, 40, 41], and to find the best routing protocol or topology to achieve the best performance in WSN. Some of these approaches are:

- Using multi-sink instead of single-sink architecture to improve connectivity
- Trying to find a method to evaluate the link behavior
- Making the routing decision a responsibility for each node

1.3 Approach Chosen to Solve the Problem

Based on all the work that has been done in the area of WSN, understanding the link behavior is the first step to develop a good topology. The current research is based on the work done in the area of the link quality and then it tries to find a good link estimator that can react quickly towards changes in the link quality. As a result, the network can adapt itself to environmental changes, mobility of the nodes and the available battery.

The second step is to develop a self-organizing algorithm that can use the information provided by the link estimator at each node to create the topology. This algorithm will allow messages to reach the base station, using the best possible link and, at the same time, balances energy consumption.

In order to achieve the purpose of this study, the research will include a practical part in which the new method and topology will be implemented using Java SunSPOTs [13]. The experiment's results are used to study the behavior of the system. This information will then be gathered and analyzed to estimate the network connectivity.

1.4 Thesis Goals and Expected Results

The aim of this research is to improve and measure the connectivity of the network as a whole. To accomplish the goal, the best topology, based on the allocation and movement of the nodes, has to be found, so that the mobile wireless sensor network (re)configures itself. This feature will allow the network to try to recover from any failure and unexpected situation that may occur, and when needed reconfigure itself based on the new circumstances. At the same time, the network has to be able to consume the minimum amount of processing power, memory usage, as well as receiving/transmitting power. These are important features to be able to keep the network alive for as long as possible.
Introduction

Therefore among the available topologies the appropriate one is chosen. Then by the means if link estimator nodes are arranged and the network is established. In order to have a reliable network, based on the important factors for network connectivity, a connectivity formula is developed and then used to calculate and report network connectivity periodically.

The network is then tested for different scenarios that may happen during a WSN life time and the connectivity is recorded and then compared to another network topology. The results are then compared to indicate that the chosen topology provides a better connectivity.
2 Background

Wireless sensor networks are used to improve the performance of several available systems, such as monitoring and controlling environmental changes. Due to their specific characteristics, like limited energy resources, low cost, small size and mobility, they become more energy sensitive and, therefore, it becomes necessary to estimate their connectivity using an energy efficient method. Each feature in WSN adds its own difficulties to evaluate the whole network connectivity. For example, mobility is a feature that can be deployed in many areas, such as target detection and gathering information of moving objects like vehicles. Meanwhile, it adds its own complexities to the network, thus creating new demands. Wireless sensor networks’ connectivity has been researched for many years, by different groups and each by considering a set of features according to their specifications. A common aspect considered by several research groups is that wireless sensor networks have a limited amount of energy; therefore, save energy, thus, prolonging network lifetime is important. Many protocols and algorithms are based on energy efficient methods for network communication [14, 15, and 16].

2.1 State of the Art

Mobility is an important factor in WSN. Several researches try to solve the up coming problems of mobility. Some of them are listed below:

- Using multi-sink instead of single-sink architecture; to get rid of the contention problem that may occur near the sink and, as a result, achieve the best performance in the network [17].

- Trying to find a method to evaluate the link behavior by using information from different layers of the protocol (cross-layer protocol), which is also used to improve the performance. After finding the quality of the links in the network, the routing decision can then be made based on these results to find the best, and the most reliable, link to use for communication [18].

- Different approaches for using mobility to help maintain the network connectivity, and increase the lifetime of WSN has been proposed in [19], including, mobile base station, mobile data collector, and Rendezvous-based solutions.

Reliability of a WSN is so important and it is affected by faults that may happen for several reasons. A WSN that is not prepared to deal with such situations may face critical problems, which lead to reduction of network lifetime or connectivity problems. There are several pieces of research about WSN reliability and fault tolerance.

- A survey presents the approaches to fault tolerance and detection techniques in WSNs [20]. The report provides a taxonomy of faults and classifies the surveyed approaches according to their ability to detect and recover from, faults. The research is based on service availability and understanding the probability of which request will lead to a valid and useful response. (Point) availability for system $A$ is defined as

\[ P(A) = \frac{MTTF(A)}{MTTF(A) + MTTR(A)} \]

$MTTF$ stands for "Mean Time to Failure" and $MTTR$ stands for "Mean Time to Repair" [21]. The research considers possible attacks on all layers of the communication that can directly influence the availability of services.
Background

- Measuring the connectivity of a network with reachability, which is defined as the ability of two nodes in the network to communicate, and it is measured based on graph theory [22].

- A comparison of the performance of different topologies of the WSN, based on different criteria like the network reliability, life time, battery consumption and latency, etc [23].

- Another method used to evaluate the link quality is studied in [24], where the link estimator is based on information provided from the physical, link and network layers in the communication protocol.

2.2 Related Work

Prolonging the network lifetime and improving its connectivity while balancing energy consumption are two important research aspects of wireless sensor networks. There is an approach in which, the author tries to develop an efficient WSN topology, based on the mesh scheme to prolong network lifetime and consume less energy. The goal is to develop the optimal network topology based on the nodes’ behavior. The paper introduced Leader-based Enhanced Butterfly (LEB) network topology, which is based on a combination of hierarchical and ad-hoc network properties. The model proposed a three-layer schema, where each layer is managed by its upper layer. The proposed LEB focused on layer two and three communications. In this method, each pair of layer-two nodes (Leader) is grouped in one cluster and it is responsible for the third-layer nodes of the whole cluster (each leader manage its own nodes and the other leader nodes). In case of low battery power, one of the leader nodes will be responsible for all the third-layer nodes and the other leader [25].

Considering the mobility of the nodes in WSN, another method that is used to balance the energy consumption is to make the routing decision a responsibility for each node based on its remaining power. Distributing the routing decision among the nodes rather than localizing it in a specific node or a base station balances the energy consumption of the network and avoids having single point of failure in the system. Besides, making all the decisions in one node will end up consuming all its energy very fast, which will lead to connectivity failures in the network as whole [26].
3 Solution Idea to be Investigated

Having in mind the purpose of this work, which is to find a model that can minimize the current problems of WSN and find an optimum pattern for node distribution while supporting reliable connectivity and route stability, the development of a good topology and understanding the link behaviour are the first steps. In order to create a routing table, each node should have information about its neighbors, the quality of the links connected to it and routing information. In addition, since nodes have a limited amount of memory and processing power, only necessary information should be stored. Important information includes neighbor information (MAC address, IP address, node distance …) and link information (link cost …), which should be updated periodically, therefore using a link estimator seems necessary. A good link estimator can provide the essential information to compose routing table.

The next step is to choose the topology and find the node distribution pattern. The topology supports all possible positions and arrangement of the nodes while it maintains network connectivity in all situations and reacts to nodes lost, and network reconfiguration, within a good time. In addition, information transmission should not be costly for the network and energy consumption should be either balanced or reduced (acceptable). Data transmission should not cause packets' collision or exhausts nodes' energy. The important factor that is considered is the mobility of the nodes. The chosen topology and link metric cover nodes' mobility.

3.1 Methodology

In WSN, use of a link estimator (metric) is a good solution to manage the neighbor nodes and then, by the information provided, it is possible to find the appropriate mechanism to propagate information among mobile nodes. This way, the network may consume less energy or a balanced level of energy; it needs fewer resources, and spends less time. As a result, the network will behave more reliably.

In a wireless sensor network, one way to develop link estimation is to use signal strength information. Signal strength by its own provides a very poor estimation. Another possible metric is channel snooping and following the sequence number of the received packets from each source. This method is very costly, because one node may listen for many packets that are not necessarily addressed to it; therefore, the lowest cost snooping method should be considered.

A good link estimator is the one that needs a small memory space and fits in a limited storage budget. It should react quickly to changes in link quality, so protocols can adapt themselves to environmental changes and mobility. In addition, it should use simple computing processes to reduce the energy costs. Furthermore, an estimator must be stable, otherwise, when the routing topology changes, it may cause a problem. A link estimator then may provide the necessary information for neighbors' management and support routing protocols.

The main subjects that had been studied within the present research are:

- Effects of nodes' mobility on the network connectivity and management: mobility of the nodes poses problems in network topology and management. In addition, since the nodes are mobile, their connectivity and reliability of the routes is another problem that should be minimized.
- Link quality: Since WSN is not stable and nodes distance may change, loss rate should be considered. Loss rate is directly affected by the distance and radio signal power. Meanwhile obstacles that may appear between the nodes can cause the signal to be either stronger or weaker and lead to signal fluctuation. In addition, environmental changes can also affect the link quality. Finding the most important entry in a stream of data is another issue in WSN, since each node may receive many packets, which are not all useful, therefore, it should be able to recognize which are the usable ones. Reducing the collision that may happen because of the nodes that are located near each other is also an important factor.

- Network topology, nodes' distribution and network connectivity: a WSN that has mobile nodes should support a high-level connectivity. Nodes distribution has direct impact on the network topology and, therefore, on handling different situations that the network may face. In addition, not all nodes in the network are suitable for data transmission and should not waste their energy passing information. Consequently not all the nodes may have a good connectivity and, therefore, an acceptable link quality. Only particular nodes that have a good connectivity measure should be involved in data communication. Hence, an appropriate topology and a good network controlling and monitoring schema can provide connectivity, reliability and lessen energy consumption, which leads to prolonging network lifetime.

3.2 Link Metric

In order to have an optimum self-organized connected network, an accurate link estimator is required. There are several challenges to define a good link metric. Mobile WSN link estimators use a combination of a number of information, consisting of: network condition, links quality, link availability and reliability, nodes specific properties (e.g. radio transmission power, storage capacity and processor power) and environmental conditions, to come up with a metric for the network. The link estimator provides further information needed for nodes' distribution and network monitoring. A good link estimator should react quickly to network changes. It should need a small memory resource and processing energy; furthermore, it should be simple to calculate.

There are several link metrics, but most of them do not cover nodes mobility in WSN. One method is Four-Bit Wireless Link Estimation [27]. In this method, a protocol independent interface for the different layers is introduced. This metric provides four bits of information: 1 from the physical layer, 1 from the link layer, and 2 from the network layer. From the physical layer, channel quality during a packet transmission is measured. From the link layer, packet's delivery and acknowledgement is measured, and from the network layer, the most valuable links for higher-layer performance are learnt. This method reduces packet delivery costs by up to 44% over other approaches and maintains a 99% delivery ratio over large, multihop testbeds.

There is another method, which is based on the shortest path algorithm, which improves the mentioned method performance [28]. In this method, each node keeps track of a set of nodes that it hears; either through packets that addressed to it or the packets snoops off the channel, and builds a statistical model of the connectivity to/from each. Therefore, to define the neighbors, an estimation of $P_{ij}(t)$ (success rate), which is more than a specific threshold, is used for each node. Each node picks as its parent the neighbor with lowest depth and link success rate above the defined threshold [28].
Solution Idea to be Investigated

Each link behavior affects the network behavior and connectivity. One poor link can decreases the network connectivity; therefore, Link estimation has a direct relation with total network connectivity. This indicates that link estimation is a derivation of successful packet delivery rate, which represents the network concentricity percentage. Therefore, in order to report network connectivity, link estimation is necessary.

3.2.1 Chosen Link Metric for the experiments

In WSN, link metric is employed to choose the most efficient path from one node to the other. Link metric replaces available algorithms for routing, like the shortest path. As is known in shortest path algorithm, a short-hop path with a poor link quality is preferred to a long-hop path with a better link quality. Such algorithms are not reliable for WSN.

In order to have a good link metric, different factors that affect the link quality of a mobile WSN should be considered. A good link metric should act quickly towards network changes, so route selection can be reasonably quick and connectivity is more guaranteed. Furthermore, this metric should be stable, so it does not affect and change the network topology continuously.

After studying available metrics, it was concluded that a good link metric for mobile WSN might consist of three factors that report the nodes situation dynamically. These factors are received signal strength indicator (RSSI), node remaining energy and received packet quality (LQI).

Since this experiment is about mobile WSN, hop count in not included in the metric. Hop count is a good indicator for mobile WSN but it is not efficient. In mobile WSN, when the number of the nodes in a link is increases the probability of packet error or lost is increases too. Therefore, hop count can reflects link reliability. However, if hop count is added to metric, then a storage space on each node must be assigned to it. In addition, each node must use extra energy to count and save the number of the hops for each received packet, therefore hop count is omitted.

Furthermore, in this experiment clustered hierarchical topology is deployed. Therefore, for level two nodes, hop count is always one, and for the rest of the nodes it is usually two.

In WSN, data transmission is done by using radio transmission power. Low-power radio signal reduces the network connectivity because it covers a small area. Meanwhile, obstacles in between have a direct impact on the radio transmission power and can decrease it. When mobile nodes are part of a WSN, their movement has impact on the signal power and link reliability, so the received signal strength indicator (RSSI) of a node is one of the factors that can be used in link estimation.

Energy is a public area of concern for mobile WSN. Since nodes are moving, it is preferred that they consume less energy in order to decrease their maintenance cost. Reducing total nodes' energy consumption prolongs network lifetime, which is a desirable attitude for WSN. Therefore, the remaining energy of a node is the next factor to include in link metric.

The quality of a packet that is received in a node is the last factor in link metric. As for all the networks, the environmental situation can modify the transmitted packets. In WSN, channel interference (Co-Channel interference) is also added to the threats that can modify a transmitted packet. The other threat that may affect the radio signal is propagation loss, which has three forms. Propagation loss can happen in the form of path loss, absorption or signal fading.
Path loss happens when a node moves far and therefore it leads to power density degradation. Absorption happens due to molecules in the earth atmosphere and signal fading happens due to weather condition.

Some of the errors that may affect the signal are path loss, absorption and fading. Therefore, the quality of the received packet is added to the link metric.

The used link metric in this research is described as:

$$\text{Link metric} = \text{Remaining energy of the node} + \text{received signal strength of the node} + \text{received packet quality}$$

To make this metric more accurate, the importance of the mentioned factors was studied, and it was concluded that according to the network topology and the node role in the network \[18\], the factors importance can be different. The link metric will be used in network topology and connectivity measurement.

WSN may face serious problems because of its nodes limited energy sources. Since this limited energy is the only resource for data storage, communication, and computation, it is always a point of attention. There had been many researches in this area. For example, \[30\], proposed a protocol, which is a trade off energy consumption and source-to-sink delay.

Since balancing energy consumption in WSN helps the network to prolong its lifetime, in this research energy is considered as one of the factors that indicate if a link is good or not. Hence, each sensor's remained energy is included in the metric formula.

WSN topology of the current research is clustered hierarchical. There are three levels of nodes in the network, and it was tried to distribute different computational jobs among all levels. This way one specific level will not be exhausted.

$$M = K1 \times Enr + K2 \times RSSI + K3 \times LQI$$

Formula 1- Link metric

$M$ is the link metric, $Enr$ is the remaining energy of the node, $RSSI$ is the received signal strength indication and $LQI$ is the link quality indication. $K1$, $K2$ and $K3$ indicating the importance of the relevant factor for the node and will be discussed in detail in later chapters.

Since clusterheads are like connectional bridge between base station and rest of the nodes in the network, so they are consuming more energy. Clusterheads have to communicate with both, base station and simple nodes. In addition, they are responsible for calculating their cluster connectivity and reporting it to the base station. On the other hand, if a clusterhead loses energy and dies, its cluster should be reconfigured and all its nodes should connect to another node. Therefore, more energy is consumed and the network may face broadcast storm. To avoid the mentioned problems more weight was putted on energy for clusterheads.

For level three nodes (simple nodes), since these nodes, are usually not clusterhead and mostly they are just sending their own information to clusterheads, at the first look signal strength is more importance. However, sometimes these nodes can act as a connection for a forth node to their clusterhead. Therefore, in this research the same weight was putted for energy and signal of the third level nodes.

### 3.3 WSN topologies

WSN can consist of thousands of nodes that each act as a source of information. Nodes are performing routing to send information to each other and they can be an information sink to receive information from other nodes or external objects. Topologies can be different,
Solution Idea to be Investigated

according to the role of one main single node or the whole network. There are three main topologies for WSN and each has its own advantages and disadvantages. The topologies are star, mesh and clustered hierarchical WSN. In order to choose an appropriate topology for WSN, main factors that should be considered are reliability, power consumption, scalability and network lifetime, transmission medium, operating environment and topology changes. In the simplest form, if there is one single main node in the network, star or mesh topologies are appropriate, while, when there is more than one main node, clustered hierarchical or hybrid topologies can be applied. However factors like nodes' density, environmental conditions (e.g. harsh environments), and radio transmission power should be considered.

3.3.1 Star topology

Star topology is one the most common network structures, mainly because it is simple to implement and navigate. Start topology consists of one central node, which is called "sink". Sink is responsible for message transmission within the network. Other nodes in the network can reach the sink with one hop using a direct link (Figure 1). The central node can be either a base station or a gateway that communicates with a base station. A Tree is an extension of star topology. It is a hierarchical star where sink nodes in each level are connected to the upper level sink node (Figure 2). In start topology, communication with the main node is direct (single-hop) and data must pass through the sink to reach its destination, therefore communication is reliable. Since the network has one main central point and nodes are independent from each other, one node failure does not affect other nodes' performance.

However, a star topology based network is not scalable and then the number of the nodes is limited. The only way to increase the network size is to improve the sink capacity, which may add load on it. Therefore, power consumption increases and it leads to an increased complexity. Numerous nodes can cause collision on the network and decrease the total performance. In addition, the network is very dependent on the sink, failure of the central node causes the network to stop working, and all the nodes will be isolated.

![Figure 1 - Star topology, Gray node: base station, Green node: simple node](image1)

![Figure 2 - Tree topology, Dark gray node: base station, gray node: simple node, parent, Green node: simple node either parent or child, Blue node: simple node, child](image2)

3.3.2 Mesh topology

In mesh networks, either each node communicates with the sink (base station) directly, or it passes multiple paths through its neighbor nodes to reach the sink. In mesh topology, each node can route other nodes' information to the sink, which means that all the nodes are not directly connected to the sink. Mesh is a multi-hop network (Figure 3).
Wireless Sensor Network Group Connectivity

In a mesh network, there are multiple redundant paths for each node around the network. If a node fails, or the link between two nodes goes down, the network automatically reconfigures itself and uses another route.

![Mesh topology](image)

**Figure 3 - Mesh topology**

In a mesh network, energy consumption of all the nodes is not the same. Nodes that are nearer to the sink consume more energy compared to nodes further away, since they are sending their own information plus the other nodes' information.

Mesh topology has the ability to reconfigure the network automatically and choose the best route by considering the link quality. Mesh networks are scalable for a medium size networks but, when it comes to big networks, they have the problem of degraded performance.

Mesh networks have a low data loss rate, but they have the problem of data latency. Depending on the distances of the nodes and the number of them, mesh network may face latency while a message goes along a multi-hop route. In addition, node density can cause the problem of overload near the base station and this leads to more latency.

### 3.3.3 Clustered hierarchical topology

Clustered hierarchical topology is a combination of tree and mesh topologies. Each cluster is a mesh network itself and it can communicate with other clusters using gateway nodes (figure 4). There is one main head node called "level 1 cluster head", in the entire network, which is responsible for communicating with the base station. Other cluster head nodes, which are called "level 0 cluster head", are responsible for the communication inside their own cluster and communication with other clusters using gateways. In this topology, communication between different clusters is not essential via cluster heads. Clustered hierarchical topology is a multi-hop mesh routing schema.

Since each cluster is a mesh, this topology can be considered reliable. It is also scalable, since new nodes can be grouped in a new cluster without any overload for other clusters. In addition, each clusterhead is responsible for managing its own cluster nodes. Compared to mesh topology, clustered hierarchical topology supports a wider coverage area.

One of the methods that help prolonging the network lifetime in a WSN is the deployment of a hierarchical structure for routing. Employing clustering algorithms can be one of the useful methods for implementing a scalable and energy efficient network, which has the main purpose of good connectivity for the whole network. The main goal of a clustering technique is to reduce the transmission distance among nodes. Therefore, employing a dynamic clustering technique can help to minimize energy consumption of the network, as communication is a major energy consumer.

There are some points that should be considered while using a clustering method, which are listed as below:
Solution Idea to be Investigated

- How clusterheads should be selected;
- Network reliability should be considered: wireless networks are usually considered less reliable than wired network because of the nature of communication medium and the nodes;
- Nodes mobility adds its own complexity to the network topology;
- Network robustness should be considered.

Figure 4 - Clustered hierarchical

3.3.4 Clustering basis

There are different clustering algorithms that can be separated from each other by the method they select clusterheads, but usually they have some basic similarities. Clustering algorithms divide a network into separate clusters and avoid clusters overlap by choosing separate clusterheads for each cluster. Approximately, in most of the clustering algorithms, non-clusterhead nodes send their data to their clusterheads and, clusterheads aggregate the received data and, send it to the sink (base station). There are several clusterhead selection examples, like:

- Linked Cluster Algorithm: node with the highest id will be selected as clusterhead.
- Enhanced Linked Cluster Algorithm: node with the lowest id will be selected as clusterhead.
- Distributed Cluster Algorithm: clusterheads will be selected according to the nodes weight.
- Weighted Clustering Algorithm: clusterhead will be selected according to nodes weight.

Each clustering method can group the nodes based on network priorities. Examples are listed below:
Load balancing is the main issue for the network. The algorithm tries to put the same number of nodes in each cluster and minimize the distance between nodes and their clusterhead; such algorithms mainly focus on Intra cluster traffic rather than external traffic. One of these algorithms is the max-min $d$-cluster algorithm.

Network lifetime is the main issue for the network. Cluster sizes are varied and clusterhead selection can be based on location, which is previously known. In these methods, inter cluster traffic is not important and the algorithm should be aware of network topology. Several approaches are proposed in this area.

Low-Energy consumption is the main issue for the network. Each node has a probability of becoming clusterhead at least once, so energy usage is distributed among all nodes. In this method, the network assumes that each node can be directly connected to the sink and the network is aware of nodes' location.

While clustering a network, there are some design principles that should be considered. Since grouping nodes into clusters uses resources for processing and communication among nodes, clustering a network has its own costs (Cost of clustering). Depending on the application that uses WSN, clusterhead selection and number of the nodes in each cluster (physical size of the cluster) is varied. In addition, data lifetime is important; in most applications real-time data delivery should be supported. A well-deployed WSN should minimize energy consumption; therefore transmission schedules should be synchronized to use minimum energy possible (Synchronization). Like any other network, WSN needs repair mechanisms. Networks may probably face link failures during their lifetime, so mechanisms for link recovery should be considered to support data reliability. Meanwhile, since more than one node may collect the same data, clustering mechanisms can be combined with data aggregation. Also, clustering design should consider efficient factors for the network to provide the best QoS, according to the demands.

### 3.3.5 Available clustering algorithms

There are several available clustering algorithms that each considers one or more factors as their basis. The first step to cluster a network is choosing the clusterheads.

- **Heuristic Algorithms**: Heuristic algorithm is based on reasonable runtime and/or finding a solution setup, as close as possible to the optimum one. Therefore, it is not based on particular metrics and it only provides reasonable performance. Some available algorithms are:
  - Linked Cluster Algorithm (LCA): an identifier (id) will be assigned to each node. The node with the highest id will be the clusterhead.
  - Linked Cluster Algorithm2 (LCA2): nodes are grouped to covered and non-covered. "Covered", means that one of the node neighbors is a clusterhead. The clusterhead among non-covered nodes will be chosen and assigned to the node with the minimum id.
  - Highest Connectivity Cluster Algorithm: connectivity is defined as the number of the neighbors. The node with the highest connectivity will be elected as clusterhead.

- **Weighted schema**: Weighted Clustering Algorithm (WCA): the algorithm is triggered periodically by an event. Whenever there is a topology change, it will reconfigure the network. It tries to provide stability. This algorithm is based on a mixture of metrics. Each node can be chosen as the clusterhead.
Hierarchical scheme: The goals of the hierarchical schema is to maximize network lifetime and distribute energy consumption, also while clusterheads selection period is running, minimize energy consumption. In addition, the hierarchical schema reduces the overhead caused by network control processes. The turning point of this method (comparing to other methods) is the clusterhead election. After the clusterheads are selected, the rest of the nodes should join the available clusters. There are different methods for grouping the nodes but mainly either the clusterheads should choose their members or it is the node's responsibility to choose a suitable clusterhead. Some available algorithms are:

- LEAC: a periodic algorithm and choosing clusterhead is based on random rotation.
- TL-LEACH: In the Two Level leach, there are two levels of clusterhead. Secondary clusterhead collects data from nodes in its related Clusters. Primary clusterhead collects data from its related secondary clusterheads.
- Energy Efficient Clustering Scheme (EECS): each node sends its remained energy to its neighbors. The node with the highest energy will be selected as clusterhead.
- Hybrid Energy-Efficient Distributed Clustering (HEED): is based on physical distance of the nodes.

3.3.6 Chosen topology for the experiment

WSN is an infrastructure-less network that consists of mobile nodes that move and communicate with each other using wireless links. Each node has a limited energy that should be used for vital activities like data transmission, request processing and necessary communication with neighbors. If this energy is used for actions, such as retransmission (due to packet loss or collision), or unnecessary communication with neighbors, then the network may loss connectivity and its lifetime decreases. In order to achieve optimum network connectivity, and to prolong its lifetime, some methods and algorithms can be used. One of the methods that can be applied on network topology to reduce energy consumption is clustering. Clustering provides a scalable network.

Self-organization is an important capability that WSN needs to present. WSN should be reliable, scalable and energy efficient during the network lifetime and self-organization plays a key role in this context. In a WSN, nodes should be organized in such a way as provide the best quality of data and use minimum energy. Since WSN is an ad-hoc network, it is not practical to group/organize the nodes beforehand. Therefore, the use of clustering mechanisms can provide solutions to overcome the problem. In clustering mechanisms, the cluster size is an important parameter since it affects the network performance. Clusters should be bounded-size with low message overhead and low overlap. Factors that should be considered while clustering a network include:

- Energy constraint: efficient use of energy is important for the node to stay alive as long as possible in the network
- Network lifetime: since nodes have limited energy and network lifetime is dependent on them, the efficient use of energy provides the benefit of extending network lifetime.
- Limited computing capabilities: sensor nodes are usually small physical objects with limited energy sources, as already mentioned, which is a fact that affects directly their processing and communicating abilities.
Wireless Sensor Network Group Connectivity

- Radio transmission power.
- Application requirements: depends on the application type, nodes distribution can be varied.

**Clustering the network**

In this experiment, clustering is chosen as the wireless backbone architecture of the network. The network is divided into three layers (three-layer architecture). In each layer, nodes have separate responsibilities, level of communication and access to other parts of the network. This architecture is flexible so it can adapt itself to topology changes of the network. There are two phases to build a clustered network. First, the clusterheads should be selected and then the rest of the nodes should be clustered. Each node can only belong to one cluster at a time and each cluster is monitored by just one clusterhead.

Network architecture is a three-layer network as below:

- Layer one: base station
- Layer two: clusterheads
- Layer three: simple nodes

Each layer has its own conditions and considerations. All the layers together help the network to have a better connectivity and performance.

**Network trend and schema**

As is mentioned above, the first phase is choosing the clusterheads. There are several algorithms, which can be used to elect clusterheads. For this experiment, the defined link metric is used to choose the clusterheads. In order to choose the clusterheads based on the metric the optimum number of the clusterheads is predefined. The optimum number of clusterhead is set according to the total number of the nodes in the network. Then, a threshold is defined to specify the minimum link metric needed for a node to be selected as a clusterhead.

When the network is started for the first time, the base station will broadcast a message to all the nodes in the network and wait for their answers. When a base station receives an answer from a node, it will check the metric of the sender node. If the metric is above the predefined threshold then that node will be chosen as a clusterhead (in this step the optimal number of clusterheads should be specified, otherwise if all the nodes are qualified to be clusterhead then the base station will be exhausted since all the nodes wants to directly communicate with it.)

After the clusterheads have been elected, each one of them will broadcast a message to all other nodes in the network and invite them to join its cluster. Clustering the rest of the nodes will be based on their link metrics.

Each node in the network may receive more than one clusterhead invitation message. At this point, there are two possible methods to cluster the network. The first way is when clusterheads are responsible for deciding if a node should belong to their cluster on not. The second method is when each node will decide which clusterhead it wants to join.

If this responsibility is assigned to clusterheads, then they need to communicate with each other and compare their information. Based on link metrics and the comparison result they will decide which node should belong to which cluster. This way the network faces delay and energy consumption. The delay is because clusterheads should wait for each other to gain complete information and build the information table. So each clusterhead should wait until other clusterheads have reached a stable situation and the network becomes converged, which
will cause latency. The energy consumption is because the propagation cost of information between clusterheads, and the comparison and calculation of energy consumption. In this experiment, it is assumed that each node, based on the received link metrics from the clusterheads, will decide which clusterhead it will join.

There may still be some nodes, which had not received a clusterhead invitation to join it. These nodes are far from the clusterheads. There are two solutions for this problem. Either each node can be set to broadcast an invitation message to all other nodes after it joins a clusterhead, or a timer can be set.

If the first solution is applied, the network will face broadcast storm, collision, latency and power consumption. So, in this experiment, it is preferred to set a timer. After the timer expires, each node will check to see if it belongs to a clusterhead. If the node finds out that it had not been assigned a clusterhead, then it will broadcast a message requesting a connection to one of the possible clusterheads. The node will receive an answer either from a clusterhead or from a secondary node. Based on the received link metrics the node will decide which clusterhead it should join. These nodes are using multi-hop links within a cluster.

Simple node properties
- The node sends its information to its clusterhead.
- Communication with the clusterhead can be either single-hop or multihop.
- The node belongs only to one clusterhead.
- The node has a table of its next hop and its' metric.
- The node can be used to connect another node to its clusterhead.

The networks should be self-organized, so in the case that the topology changes, it can reconfigure itself. To have a well built self-organized network, different responsibilities within the network are categorized and assigned to different level of nodes.

Base station responsibilities
- The base station should monitor the network, calculate, and report network connectivity, based on the information it receives form the clusterheads.
- The base station is responsible for choosing clusterheads.
- When the network is converged, the base station checks the number of the connected nodes. If a node is missing, it is because of its distance from all the clusterheads. In this situation, first, the base station should use its maximum transmission power to connect to the node and, then it sends a message to the node and tells it to come closer. Since this method consumes base station energy, it is not implemented on the current research.

Clusterhead responsibilities
- The clusterhead is responsible for intra-cluster communications. If a node is about to leave the cluster and lose connectivity, the clusterhead should send a notification message and tell the node to come nearer.
- To aggregate information (link metrics) within the cluster and sends it to the base station, for connectivity calculation.
- Each clusterhead has a table of its nodes and their metrics.
If a clusterhead fails to operate, first the base station should try to avoid it by sending a message to the clusterhead and so prevent disconnection. However, if the clusterhead is lost, only the nodes within the disconnected cluster should start broadcasting a message, requesting a connection to one of the other possible clusterheads. The node will receive an answer either from a clusterhead or from a secondary node. Based on the received link metrics, the node will decide which clusterhead it should join. This solution prevents the entire network reconfiguration since it wastes energy and time.

In this topology, communication between clusterheads is prevented to save time and energy and to avoid collision. It is also assumed that, if a node decides to leave the cluster and the clusterhead cannot prevent it, the node itself will start sending messages to check which new cluster it should join (the decision will be based on link metrics).

### 3.4 Network Connectivity

While in fixed infrastructure wireless networks, it is sufficient that each mobile node has a wireless link to at least one other node. Connectivity in an ad-hoc mobile WSN is complicated. In mobile WSN, node density and transmission ranges are two important factors that affect network connectivity. One idea to implement connectivity measurement on WSN can be briefly explained below:

Assuming that there is a random distribution method for nodes and a link model, the purpose can be finding out the number of the nodes needed to cover the whole network, with a desirable connectivity (K-connectivity) degree for the maximum distance (r0) in WSN. To achieve a fully connected network, there must be a wireless multihop path from each mobile node to another mobile node.

In WSN, one of the properties that should be adjusted correctly is the transmission power of nodes. Transmission power has a direct relation with the coverage ability of a node. Transmission power can reduce interference, provide QoS and control topology.

**Minimum radio transmission range (r0) needed for N nodes in Area A**

To measure the connectivity in a WSN with density \( \rho \), and minimum \( n_0 \) neighbors for each node, the minimum transmission rate for each node should be recognized to avoid isolated nodes and provide full connectivity.

The transmission range should cover the network in a way that there is no isolated node (a node without any link to other nodes) and the network is fully connected (each node can access other nodes through a multihop path). Besides the above-mentioned aspects, a network should be designed in a way that it always requires minimum communication resources and, in the case that a network node failed to operate, the network remains connected. Meanwhile, the mobility of the nodes and its effect on the network should also be considered. Within the design, the minimum number of neighbors that each node should have will be known.

To answer the above-mentioned questions, firstly the distribution model of the nodes should be defined, then the paths between nodes should be recognized and, finally, the mobility pattern of the nodes should be explained.

**Avoiding isolated nodes**

One of the factors that should be considered is the probability of isolated nodes, which can cause the network to stop working properly. It can be concluded that each node needs a certain number of neighbors to avoid being isolated.
Solution Idea to be Investigated

**Minimum Node Degree**

To have a fully connected network, each node should have a minimum number of neighbors, which is called "minimum node degree". In order to find the Minimum Node Degree, the required radio transmission rate should be calculated.

The current research tries to improve wireless multihop mobile network connectivity.

3.4.1 Graph theory

In order to maximize the network lifetime, the WSN nodes behaviour should help to minimize the energy consumption. To estimate the network connectivity and support mobile nodes connectivity, graph theory is applied. To represent the network, the graph concept is used. The network can be defined as follow:

\[ WSN = G(N, E) \]

**Formula 2-Graph Formula**

Where \( N \) is set of the nodes in \( WSN \) and \( E \) is wireless links in \( WSN \).

Node degree: is either number of the neighbors of the node, or number of the links of the node.

Connected network: for every two nodes, a path is available; otherwise, it is a disconnected network

K-connected WSN: when, for each two nodes, there is at least one path to connect them, and there is no node, the removal of which would cause the network to be disconnected, then the graph is k-connected.

3.4.2 Connectivity Formula

The measure that determines how good an ad-hoc network is connected is based on the probability of communication between any two nodes in the network, whether this communication is single hop or multiple hops from the source to the destination.

Assume that the network is represented by graph \( G (N, E) \). \( N \) represents the set of all the nodes in the network, and \( E \) is the set of all the edges or links, where an edge exists between two nodes in the network, only if the two nodes are in the range of each other.

Two nodes in the network, \( n_i \) and \( n_j \), are neighbors if the distance \( (d_{ij}) \) between the two nodes is smaller than the transmission range of the nodes \( (r) \). Assuming the network as homogenous, the transmission rate \( (r) \) is the same for all the nodes.

Connectivity in graph theory consider the probability \( (P_{ij}) \) of two nodes \( n_i \) and \( n_j \) to be 1, which means that they are connected if the nodes are neighbors or if there is a specific number of nodes between them that makes them connected through multiple hops. Since a graph can be considered to be bidirectional, if node A can reach node B in the network, then node B can also reach node A.

A way for measuring the connectivity is to measure if every pair of nodes in the network is connected by either a single hop or multiple hops, which is what is called the "reachability" in graph theory. This is measured by the following formula [22].
Wireless Sensor Network Group Connectivity

\[ R = \frac{\text{number of connected pairs of nodes}}{\binom{N}{2}} \]

**Formula 3- Basic reachability formula**

\( R \) is the network reachability and \( N \) is the entire number of nodes in the network.

This method of connectivity measurement only measures the availability of a link between two nodes in the network. Therefore, for the whole network the connectivity value will range between 0 and 1, where 0 implies that there are no nodes connected, and 1 implies that all the nodes in the network are connected, and if they can reach each other. However, this formula does not give a good indication about how good and reliable the network is, since being able to reach all the nodes does not mean that the available link is good and it is not trusted if the received information is not corrupted or altered due to fading in signal strength.

As for the studied case, the aim is not only to know if two nodes can only reach each other, but how well the network is connected and how well the link that connects the base station to each node in the network is, so the information about the existence of the link is not enough. It is important to know how good the links are. For this reason, weights were added to each link in the network, which will be measured by using a link metric. The same link metric is used to set up the network clusters and to make each node decide to which cluster it should be connected.

Considering what was highlighted above, the best way to measure the connectivity of the network considered in this work is by using the reachability formula, but with an addition of the weight for every link, calculated by the link metric \((M)\), so the formula for the connectivity will measured as follows\[^{[22]}\]:

\[
C = \frac{\sum MQ}{N(N-1)} \times i=1 \frac{Mi}{Q}
\]

**Formula 4- Connectivity formula**

\( C \) is the network connectivity, \( N \) is the set of all the nodes in the network whether they are connected or not, \( Q \) is a subset of \( N \) that consists of all the nodes that are actually (physically) connected and \( Mi \) is the metric for link \( i \).

Using formula 4, it is possible not only evaluates the network connectivity, but also how reliable the connections are.

Although this formula does not accurately show the situation of a particular link in the network. The above-mentioned formula is just used to have a scale to compare the results of the test in this work. In order to see the results of a link behavior in the network the formula should be improved. Therefore, if a link gets weak or it fails to work in the network the connectivity drops more sharply and the base station will be notified.

In order to make this measurement more accurate to show the effect of one link failure more specifically, the concept of RSS is used. RSS stand for square root of the sum of squares and it is used to calculate the aggregate of the accuracy of a measurement when the accuracy of all measuring components is known. In formula 4, the accuracy of each node is known. Therefore, RSS can be applied for the metric calculation of the final connectivity formula. Therefore, the formula for the connectivity will measured as follow:
Solution Idea to be Investigated

\[ C = \frac{Q(Q - 1)}{N(N - 1)} \times \sqrt{\frac{Q \sum (Mi)^2}{Q}} \]

*Formula 5 - Connectivity formula to measure link behavior effect*

Another way to improve the formula is to calculate the importance of each cluster and the relative link from the base station to the clusterhead. The calculated value can be multiplied by formula 4.

Since each clusterhead is responsible to manage its cluster then it can be implied that the link from clusterhead to the base station is more important for network connectivity.

To measure the cluster connectivity, two types of links are considered. Links from clusterheads to the base station and links inside the clusters. These links are either between clusterhead and a simple node or between two simple nodes.

The quality of the link from clusterhead to the base station is calculated from the link metric formula used for choosing clusterhead and is shown by \( M_{ch} \). In addition, the number of the nodes in the cluster is important, as it can be an indication for the importance of the cluster.

As an example, if cluster "A" is an empty cluster or if it has one node inside it and cluster "B" has 5 nodes, then cluster "B" is more important for the network. If the link from cluster "A" to the base station is removed then the network will loose two nodes, while if the link from cluster B to the base station is removed then the network will lose six nodes, so the connectivity decreases more. Therefore, the number of the nodes in the cluster is important and it should be added to the formula. From the above discussion, it can be concluded that a cluster quality is calculated by:

\[ CHQ = M_{ch} \times \frac{1}{Nm} \times \sum_{i=1}^{Nm} Mi \]

*Formula 6 - Cluster quality*

\( CHQ \) is the connectivity of the cluster, \( Nm \) is the number of the nodes inside the cluster, \( Mi \) is the link metric for node \( i \) in the cluster and \( M_{ch} \) is the metric for the clusterhead to the base station link. For all the clusters in the network the clusters quality is:

\[ NQ = \frac{1}{N} \sum_{m=1}^{P} \left( M_{ch} \times \sum_{i=1}^{Nm} Mi \right) \]

*Formula 7-Total network clusters quality*

\( NQ \) is the total network quality and \( P \) is the number of clusterheads in the network.
Detailed Description of the Investigated Solution

4 Detailed Description of the Investigated Solution

4.1 Physical or Hardware Structure

The hardware that was used to perform the practical part of the research is Sunspot. Sunspots are small programmable sensors, developed by Sun Microsystems. These devices can interact with each other and with their environment, thus they can be used in WSN implementation. Each Spot consist of [35]:

1- Main board that contains:
   a. A processor: 180 MHz 32 bit ARM920T core.
   b. Memory: 512K RAM - 4M Flash.
   c. Power management circuit: responsible for charging the battery, controlling the power for the processor board and sensor board and powering the LEDs.
   d. Radio transceiver and antenna: IEEE 802.15.4 operates at 2.4 GHz.

2- eDemo Board that consist of:
   a. Light sensor: light to voltage sensor, where the output of the sensor is in the range 0.1V to 4.3V, which represent dark to light range.
   b. Temperature sensor: internal temperature sensor able to sense temperature between -40 to +125 degree.
   c. LEDs: eight tri-color (red-green-blue) LEDs.
   d. Switches: 2 push buttons which can be used as an input to the board.
   e. An accelerometer.
   f. Atmega88 processor that controls the LEDs and provides multifunction input/output pins.

3- Battery: 3.7V 750 mAh rechargeable lithium-ion battery; it has a protection circuit to protect it from under voltage and over charge conditions, and it can be charged via a USB connector but it cannot be replaced by a battery of another type.

Figure 5 - Sunspot
Sunspots use Java Platform Micro Edition (Java ME), which is a platform used for embedded systems and mobile devices, and it uses Squawk as its virtual machine. Since Java can be independent from the hardware, all the Java applications that are developed to work on the sunspot work without an operating system. Programming the Sunspots can be done using any Java IDEs like Netbeans or Eclipse.

Sunspots use a radio transceiver, which is the TI CC2420 chip (ChipCon), this chip, is 2.4 GHz IEEE 802.15.4 compliant used for wireless communication applications that need low energy consumption.

The used link metric takes into account the RSSI and the received packet quality, which are both built in and calculated by the CC2420 chip in the Sunspot based on IEEE 802.15.4 as follow:

\[ P = RSSI_{\text{VAL}} + RSSI_{\text{OFFSET}} [\text{dBm}] \]

**Received Signal Strength indicator, and Energy Detection:**

RSSI is an estimate of the power of the received signal, and it is supposed to be used for choosing the best channel in the channel choosing algorithms. CC2420 has a built in RSSI, it gives a value of eight bits (RSSI.RSSI_VAL) and it indicates the strength of the received signal, this value is the first eight symbols following the SFD (start-of-frame delimiter)\(^{[33]}\).

The RSSI value is averaged over 8 symbol periods (128 µs), and the CC2420 chip returns status byte during receiving packets, one of the bits is RSSI-VALID, which returns 1 if the RSSI value is valid which means that the receiver was enabled for eight symbol periods.

The RSSI register value (\(RSSI.{RSSI\_VAL}\)) can indicates the power (\(P\)) at the radio frequency pins as in the following equation \(^{[34]}\):

\[ P = RSSI_{\text{VAL}} + RSSI_{\text{OFFSET}} [\text{dBm}] \]

**Formula 8 - Power at radio frequency pins**

Where \(RSSI\_OFFSET\) is measured by testing and experiments during the development of the system, and it is approximately -45 dBm, and so if the \(RSSI\_VAL\) is equal to -20 dBm then the RSSI register value will be -65 dBm.
Link Quality Indication:

The LQI is an indicator of the quality of the received packets, the measurement of the LQI is done based on the received signal strength indication, the signal-to-noise ratio or a combination of both methods together\(^{[33]}\).

Measuring the LQI should be done for each packet that is received, and it is the software responsibility to generate the value of the LQI to the application.

Using the RSSI to calculate the LQI does not always give accurate result and it has some disadvantages and for this reason the CC2420 provides an average correlation value for each packet base on the first 8 symbols after SFD (start-of-frame delimiter)\(^{[34]}\).

Therefore, when a frame is sent the RSSI value along with the average correlation and the CRC (cyclic redundancy check) are appended to the frame, and they are used to calculate the value of the LQI.

4.2 Logical or Schematic Structure

4.2.1 Algorithm description

In the experiments performed in this work, each node can have one of the three available states. These states specify the node role and connectivity situation on the network.

<table>
<thead>
<tr>
<th>State number</th>
<th>State description</th>
</tr>
</thead>
<tbody>
<tr>
<td>One – 1</td>
<td>Identification of the base station. Only base station is marked with &quot;one&quot;. This specification is used by other nodes when they receive a packet.</td>
</tr>
<tr>
<td>Two – 2</td>
<td>The node is a clusterhead and it is directly connected to the base station</td>
</tr>
<tr>
<td>Three – 3</td>
<td>The node is a simple node. It is either directly connected to a clusterhead using a single hop connection or, it is connected to another node with the specification of number &quot;three&quot;.</td>
</tr>
<tr>
<td>Four – 4</td>
<td>The node is not connected. This node broadcasts a message to other nodes in the network and asks them to join their cluster.</td>
</tr>
</tbody>
</table>

Table 1 - Nodes Specification

In addition, to control the events on the network and give them an order, different timers are used. Furthermore, timers start working when the related node, receives a special message, notifying that it enters in a new situation.

When the network starts working, all the nodes are specified as four, which means that they do not belong to any cluster. The base station will start broadcasting a message, informing all the nodes, that the network is started and asks them to send their metric for it. Each node that receives this broadcast message from the base station will calculate its own metric, based on the defined formula and, sends this information as a unicast packet to the base station. The base station will receive and then rearrange the received metrics, choosing the best ones and ordering the nodes' addresses and their metrics on the clusterhead information table. Then the base station will send a unicast message to the chosen nodes and notify them that, they had been chosen as a clusterheads. Each node that receives this unicast message from the base station will be set as a clusterhead. The specified nodes' specification will change to "two" accordingly, indicating that it is a clusterhead. After a specific elapsed time, the clusterheads
will start broadcasting a message to all the nodes in the network and asks them to join their clusters. This time is needed for the network to identify all its clusterheads. Each node may receive many packets from clusterheads; they will check the received metrics and choose the best one. The node will then send a unicast message to the address of the clusterhead they had chosen, containing its address and metric. The node specification will automatically changes to three, showing that the node is connected to a clusterhead.

<table>
<thead>
<tr>
<th>Timer place</th>
<th>Timer description</th>
</tr>
</thead>
<tbody>
<tr>
<td>First timer</td>
<td>4 sec – sets on the base station at the beginning of the program to wait, then receives the packets from all the nodes. This is done in order to choose the clusterheads.</td>
</tr>
<tr>
<td>Second timer</td>
<td>5 sec – sets on the base station, after it notifies the clusterheads, it will wait 5 sec and then asks them to send their clusters' information (metric).</td>
</tr>
<tr>
<td>Third timer</td>
<td>2 sec - after 2 sec, the base station will start calculating the connectivity.</td>
</tr>
<tr>
<td>Fourth timer</td>
<td>Each 5 sec, the simple nodes send their information to the upper level node, in order to be monitored.</td>
</tr>
<tr>
<td>Fifth timer</td>
<td>Every 10 sec, the clusterheads send their information to the base station, to calculate connectivity.</td>
</tr>
<tr>
<td>Sixth timer</td>
<td>This timer will be set on the nodes, as soon as it receives the first &quot;come back&quot; message.</td>
</tr>
</tbody>
</table>

Table 2 - Timers

A timer is set on all the nodes. After this timer expires, the node will check its specification. If it is still four, the node will find out that it is not connected to any node in the network. Therefore, it will start broadcasting a message, to all the nodes in the network and inform them that it wants to join them.

Each node on the network that receives this broadcast message first checks its own specification. If it is "two" or "three", it means that the node is connected. Therefore, it will send a unicast message for the sender node, containing its address and metric. The sender node will check all the received metrics and choose the best one to join.
Detailed Description of the Investigated Solution

Figure 7 - Network establishment flowchart

Start

- All nodes’ specifications are 4 except base station is 1
  - Base station broadcast a message to all the nodes asking for their metric and wait for the answers to decide the cluster heads

All the nodes that receive the message from the base station will calculate their metric based on the message they had received and send it as unicast to the base station

- Base station will received the metric from the nodes and arrange them based on the best metric
  - The base station will chose the nodes with the best 2 metrics it received to be clusterhead and it will add their information (address and metric) to the table of the clusterhead
  - The base station will send a unicast to the clusterhead to tell them that they are clusterheads

- The nodes that received this message will change their specification to 2 and become clusterheads
  - The clusterheads then will broadcast a message to all the nodes in the network with specification of 4 and ask them to join a cluster

- The nodes will receive the broadcast from the clusterheads and will calculate their metric to each clusterhead and decide based on the best metric to which cluster they want to join
  - The node will change its specification to 3, and send a unicast message to the clusterhead that it chose to join with its metric and address

- The clusterheads will receive a message from the nodes that they decided to join their cluster and it will add the nodes’ information (address and metric) to its table of nodes

Network Established
4.2.2 How $K$ value in the link metric formula is assigned for each node role

In order to find out the appropriate $K$ value for each parameter in formula 1, five sets of conditions were tested. For each condition network connectivity was measured 10 times and the average was calculated. Then, based on the test result the best set was chosen. To make the experiment more accurate, the average remaining battery of the nodes was also considered. Therefore, the best set was chosen as the one which supports the best connectivity, while consuming less energy than two others.

First experiment: $K_1$ and $K_2$ are set to 0.45 and $K_3$ is set to 0.1. $K$ values are the same for both clusterhead and simple node metric formula. The network average connectivity was 0.7237 and the average battery was 0.8725.

Second experiment: $K_1 = 0.6$, $K_2 = 0.3$ and $K_3 = 0.1$. $K$ values are the same for both clusterhead and simple node metric formula. The network average connectivity was 0.7796 and the average battery was 0.8725.

Figure 8 - Network connectivity using set one for metric

Figure 9 - Network connectivity using set two for metric
Detailed Description of the Investigated Solution

Third experiment: $K_1 = 0.3$, $K_2 = 0.6$ and $K_3 = 0.1$. K values are the same for both clusterhead and simple node metric formula. The network average connectivity was 0.7038 and the average battery was 0.8725.

Fourth experiment: $K_1 = 0.6$, $K_2 = 0.3$ and $K_3 = 0.1$, for the clusterhead metric formula and $K_1 = 0.3$, $K_2 = 0.6$ and $K_3 = 0.1$, for the simple node metric formula. The network average connectivity is 0.7345 and the average battery is 0.8725.

Fifth experiment: $K_1 = 0.3$, $K_2 = 0.6$ and $K_3 = 0.1$, for the clusterhead metric formula and $K_1 = 0.6$, $K_2 = 0.3$ and $K_3 = 0.1$, for the simple node metric formula. The network average connectivity was 0.7295 and the average battery was 0.8725.
After comparing the average connectivity for different sets of metric, it was concluded that set two has the best connectivity, and the second candidate, is set four (figure 13). Set two has decreases sharply when the energy drops down; therefore it is very dependent of energy parameter (figure 14). Considering that in mobile WSN, the only source that keeps the network active and connect, is its nodes energy, set two is not an appropriate choice. Replacing nodes battery is not an efficient way to keep the network active.

On the other hand, the main goal of this experiment is deploying a topology that provides the best connectivity, while it is reliable and the network is stable. In addition, the network should consume less energy. Therefore, set four had been chosen for metrics' "K" values of this experiment.
4.2.3 Link metric usage

In the current study, the link metric has three usages and, its accurate formula is based on the nodes' role on the network. Below is a description of the events that use link metric.

Sunspots RSSI range is from -60 to 60 (dbm), while in the real experiment the maximum and minimum received RSSI are -30 and 30. To use this range in the formula, it is scaled as below:

\[
RSSI = \frac{((RSSI + 30) * 100)}{60*1000}
\]

Formula 9 - Packet RSSI

Sunspots Maximum LQI is 255. In order to scale it and use it on the metric, it was calculated as below:

\[
LQI = \frac{((LQI * 100))}{255* 100}
\]

Formula 10 - Packet LQI

4.2.4 Clusterhead selection

Based on the received metric from all the nodes in the network, the base station chooses the clusterheads. The received metric in this step is:

\[
M_{ch} = 0.60*E_r + 0.30*RSSI + 0.10*LQI
\]

Formula 11 – Clusterhead selection link metric

\(M_{ch}\) is the clusterhead selection metric. \(K1\) shows the weight of the energy component in the formula (\(E_r\)). Remaining energy of a node is an important factor, especially if the node is a clusterhead. Clusterheads are responsible for calculating their clusters' connectivity and sending it to the base station. In addition, they are responsible for monitoring their cluster, so they need more energy compared to the rest of the nodes in the network. In the current experiment, \(K1\) is set to 0.60.
Wireless Sensor Network Group Connectivity

*K2* shows the weight of the *RSSI* (Received Signal Strength Information) for each node. In the current experiment, *K2* is set to 0.30.

*K3* shows the weight of the *LQI* (Link quality Information) for each node. *LQI* indicate the probability of error for a received packet. In the current experiment, *K3* is set to 0.10.

The formula factors in this step are:

*En*: the remained energy of the clusterhead.

*RSSI*: the RSSI of the sent packet from the base station to the clusterhead.

*LQI*: the LQI of the sent packet from base station to the clusterhead.

Each node calculates its own metric and sends it as a number to the base station. In the current experiments, this calculation is distributed among the nodes, in order to prevent the base station energy depletion, due to the overhead in performing the calculation for all nodes.

After all nodes had sent their metric to the base station, the base station chooses the best metrics and, informs the related nodes, that they are the clusterheads.

### 4.2.5 Clusters shaping (development, forming, arrangement, configuration)

After the clusterheads had been chosen, they will broadcast their own metric and address to all the nodes in the network. Each simple node that receives this message checks the metric and compares it to the other received metric. Then it will choose the best metric and sends its sender (clusterhead) a message, indicating that the node joins it. The sent information from the clusterheads is their remaining battery. The receiver node will calculate the clusterheads' metric by using the metric formula replacing the elements as:

*En*: the received energy from the clusterhead.

*RSSI*: the RSSI of the received packet from the clusterhead.

*LQI*: the LQI of the received packet from the clusterhead.

K value here is:

*K1*: is set to 0.30.

*K2*: is set to 0.60.

*K3*: is set to 0.10.

\[
M_n = 0.30 \times En_r + 0.60 \times RSSI + 0.10 \times LQI
\]

*Formula 12-Joining clusters link metric*

*Mn* is the node link metric for joining a cluster. Each node calculates it's metric by itself and sends it as a number to the chosen clusterhead. In the current experiments, this calculation is distributed among the nodes, in order to prevent the clusterheads energy depletion, the same way as mentioned above.

After all the nodes had sent their identification to the clusterheads, the clusterheads will keep on sending them periodic messages and, ask them to send their new remaining battery in order to calculate the metric, monitor the nodes and calculate the cluster connectivity.

### 4.2.6 How connectivity is calculated

In the current experiment connectivity calculation is distributed among the nodes, clusterheads and base station.
Detailed Description of the Investigated Solution

Each node is responsible for sending its' remaining energy to its upper level node. If the upper level node is not a clusterhead, then it may receive several packets from its' other nodes, containing their remained energy. For each packet that it receives, it calculates the metric separately, using the received packet RSSI and LQI and the energy sent to it. Then the node calculates the average metric of its nodes and sends it to its upper level node. In addition, it will send its own remaining energy, so the upper level node can calculate their link metric too.

Next level is the clusterhead. Each clusterhead receives packets from all its nodes, containing the node remaining energy. For each received packet, it calculates the metric using the received energy and the received packet RSSI and LQI. Meanwhile, if each of these nodes is parent to another node, it also sends its child metric to the clusterhead as well. Clusterheads calculate the average metric of their cluster and send it to the base station. In addition, they send their own remaining battery to the base station. The base station receives to types of packets. One is the clusters metric and the other is the clusters remaining energy. The base station calculates the "clusters to the base" link metric using the received battery amount, LQI and RSSI. Then it calculates the whole network connectivity based all the metric and by using the defined connectivity formula.
Wireless Sensor Network Group Connectivity

Figure 15 - Connectivity procedure

- The base station waits a specific time before declaring the establishment of the network
- The base station sends a message to all the clusterheads that the network is established

- The clusterhead will receive the message from the base station and they have low cases

Does the clusterhead has nodes in its cluster?

Yes

- The clusterhead will send a message to the nodes in its cluster that the network is established and they should start sending their information

- The nodes will receive the message of the establishment of the network
- The nodes will send their battery to their clusterhead every 5 seconds

- The clusterhead will receive the battery of the nodes in its cluster
- The clusterhead will calculate their new metric and replace the old metric in the node table with the new one
- The clusterhead will calculate the connectivity of its cluster and send it to the base station with the number of the nodes in its cluster
- The clusterhead will send its battery to the base station

- The base station will receive the new connectivity of each cluster and the battery of the clusterhead and use them to calculate the new final metric of the cluster
- The base station will then replace the old metric of the clusterhead with the new one

No

- The clusterhead will send its battery to the base station every 5 seconds

- The base station will receive the battery of the clusterhead, calculate the new metric and replace the old metric with the new one in the table of the clusterhead

- The base station will calculate the connectivity of the network

End
4.3 Behavioural or Functional Description

In mobile WSN, network infrastructure is not fixed and nodes can move. However nodes' movement should not affect networks' connectivity. In order to have a well connected network and preventing nodes to be lost, there are some mechanisms described in the literature \[25, 32\]. Since balancing energy consumption is one of the main goals of this experiment, therefore, node monitoring and management is distributed among all the nodes in the network.

In this experiment, base station is responsible for monitoring and managing clusterheads. For rest of the nodes in the network, if a node is directly connected to a clusterhead, using a single hop connection, it will be managed by its clusterhead, whereas if a node is connected to a clusterhead using a multi hop connection, it will be managed with its own upper level node (which is a simple node).

To have a better understanding of the nodes situation, considering the node role and connectivity situation on the network, the available array of the LEDs in the Sunspot's platform is used, being set with different colours according to the description provided in Table four.

<table>
<thead>
<tr>
<th>LED Colour</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yellow</td>
<td>The node is a clusterhead and it has at minimum one node in its cluster. Node specification is &quot;two&quot;.</td>
</tr>
<tr>
<td>Green</td>
<td>The node is a clusterhead but it does not have any node in its cluster. Node specification is &quot;two&quot;.</td>
</tr>
<tr>
<td>Light Pink</td>
<td>The node is a simple node and some other simple node(s) is (are) connected to it. This node acts like a &quot;bridge&quot; for the nodes connected to it and sends their information to its own clusterhead. Node specification is &quot;three&quot;.</td>
</tr>
<tr>
<td>Blue</td>
<td>The node is a simple node and no other node is connected to it. This node is either directly connected to a clusterhead or it is connected to another simple node. Node specification is &quot;three&quot;.</td>
</tr>
</tbody>
</table>

Table 3 - LEDs colors

4.3.1 Scenario 1: a node going far from the network

In this scenario, nodes are programmed to send their remaining battery periodically, to their upper level node, which is responsible for their control.

Each simple node in the network, sends its' remaining battery level periodically, either directly to its clusterhead or to an upper level simple node, which it is connected. Either way, the upper level node is the one who is responsible for this node management. Each time a node remaining battery is received, the responsible upper level node, will calculate the link metric from the node to itself and, then check the calculated metric, to see if it is less than the predefined threshold metric of the network or not. If the metric was less than the threshold, the upper level node will send a notification message, for three times to the node and tells it to get nearer. In the current experiment the node will blink three times. After three times, the upper level node checks the metric again and if it was still less than the threshold, then it will send a message to the node, suggests it to "choose another node to connect". When the node receives this message, its specification will be changed form "three" to "four", indicating that
it needs to be connected to another node. In this step, the node starts broadcasting a message to all the nodes in the network, requesting to join one. All the other nodes in the network receive this broadcast message, and they will answer it by sending their address and metric. The node will then choose the one with the best metric and, connects to it by sending a unicast message containing its address and metric.

Meanwhile a timer will be set on the node when it receives the third "come back" message. If this timer expires and the node had not received any message form its upper level node, it assumes that it is not connected, so its specification automatically changes to four, indicating that the node should start broadcasting a message and request available clusters to join them.

Figure 16 - Senario one procedure
Detailed Description of the Investigated Solution

4.3.2 Scenario 2: a clusterhead going far from the network

In this scenario, clusterheads are programmed to send their remaining battery and their clusters' metric periodically to the base station.

Each time the base station receives the information of one clusterhead, it will calculate the metric and check it, to check if it is less than the predefined threshold metric of the network or not. If the metric was less than the threshold, the base station will send a notification message, for three times to the clusterhead and tells it to get nearer. In the current experiment the clusterhead will blink three times and. After three times, the base station checks the metric again and if it was still less than the threshold, then it will send a message to the clusterhead, indicating that it is not suitable to be a clusterhead. When the clusterhead receives this message, its specification will change form two to four, indicating that it needs to be connected to another node. In this step, the node starts broadcasting a message to all the nodes in the network, requesting to join one. All the other nodes in the network receive this broadcast message, and they will answer it by sending their address and metric. The node will then chooses the one with the best metric and, connect to it by sending a unicast message containing its address and metric.

Meanwhile a timer is set on the clusterhead when it receives the third "come back" message, if the timer expires, and the clusterhead had not received any message form the base station, it assumes that it is not connected, so its specification automatically changes to "four", indicating that it had changed to a simple node and should start broadcasting a message, requesting available clusters to join them.

A question arises here, which is: what will happen to the clusters' nodes while the clusterhead is not connected to the base station or it is far from the base station?

There are two available solutions considering the current topology. The first solution is to notify all the nodes in the cluster of the new situation. This notification should be either sent by the clusterhead itself or the base station. The nodes specification should change to "four", showing that they are not connected. Then they should all start sending broadcast messages to all the nodes in the network and ask them to join their clusters. This way the network will face a broadcast storm, both sender and receiver nodes are consuming energy, the network connectivity is not good since there are many disconnected nodes and, there are many lost nodes. Therefore, this solution was not implemented.

The second available solution is to keep the nodes connected to the clusterhead. Since the network is a mobile WSN, all the nodes are moving alongside their clusterheads to be connected to it. In addition, even when the clusterhead is about going far from the base station, it is monitoring its cluster, therefore, the nodes are still connected to the clusterhead. Hence, when the clusterhead connect to another node, it still has all its cluster information and it is not necessary to waste energy and time to reconnect the nodes. This way network and clusters reconfiguration is prevented.

It should be mentioned that the idea of using a back up node as been studied[25] as well. This way either the nodes in different clusters should be able to communicate with each other or each clusterhead should choose another clusterhead or node as its backup. If the nodes in different clusters are communicating with each other, the network topology will be half mesh, which is not supported in this experiment. Otherwise, the clusterheads should be set to choose a back up node (another clusterhead or a simple node in another cluster); this selection should be base on link metric. Since nodes are mobile in the network, history is not reliable. Therefore, the idea of a back up node is not a good solution.
Figure 17 - Scenario two
Detailed Description of the Investigated Solution

4.3.3 Scenario 3: a node/clusterhead dies
When a node dies, it can be because of different reasons. One reason can be that the node battery has finished. Other reason is that the node had gone faraway form its parent and it had not comeback nearer. In either of these situations, the node specification will change to four and the node will rejoin the network. If a node battery changes, the node will be restarted and it acts as a new node that wants to join the network. On the other hand, if a node looses connectivity to its parent, it is set to change its specification to four, after it receives the "Choose another clusterhead" message from the parent. When the node receives this message, its specification dynamically changes to four and the node will act as new node which wants to join the network. In order to handle the exception that may happen because a node fast movement (which may lead to connectivity lost before receiving the "Choose another clusterhead" message or the specified number of the "Come back" message), a timer is defined on each node. The first time a node receives a "Come back" message, this timer will start working. If the timer expires, the node specification will automatically changes to four and the node will start broadcasting and asks other clusters to join them.

4.3.4 Scenario 4: addition of a new node
When a new node enters the network, its default specification is "four", which means that it is not connected to any node or clusterhead in the network. The new node will start broadcasting a message containing its address and, indicating that it wants to join a cluster. All the nodes in the network will receive this message and, answer it as a unicast message by sending their address and metric. The node will decide which cluster it should connect, based on the received metric. It will choose the best metric and send its own metric and address for the chosen node. Automatically new nodes specification will change to "three", indicating that it is connected to an upper level node and stops it from broadcasting. Then, the parent will send a message to the node and, inform it to send its remaining battery every five second to the parent.

4.3.5 Scenario 5: minimum number of clusterheads are reached in the network
Minimum number of the clusterheads is defined for the network. The base station periodically compares the available number of the clusterheads with the predefined minimum number. When there are less than the specified numbers of the clusterheads in the network, the base station broadcast a message to all the nodes in the network directly and asks them to send their metrics to it and change their specification to four. At this point the network reconfigures itself. The base station will choose the best metrics and choose the related nodes to be the new clusterhead. The clusterheads will broadcast a message to all the nodes in the network and asks them to join their cluster. All the nodes will join the appropriate cluster based on the best calculated link metric.
A new node enters the network

- The node starts broadcasting a message that it wants to join the network every 5 seconds.
- The node has a specification of 4

- All the nodes in the network that receive the broadcast from a node that wants to join the network will respond with their metric by a unicast message

- The new node will receive all the metrics that were sent to it from all the nodes that received the broadcast request it sent
- It will compare the metrics and chose the node with the best metric and join its cluster
- The new node will send a unicast message to the node it has chosen and tell it that it has chosen to connect to its cluster

- The node that received a message that the node decided to join its cluster will add the node to its table of information about the nodes
- The node will then send a message to the new node and ask it start sending its battery every 5 seconds

The new node has joined the network

Figure 18 - Scenario four
Detailed Description of the Investigated Solution
5 Test of Solution

In order to test the suggested method, the network was established 36 times and its connectivity was reported. In addition, to check the quality of the proposed metric, the test was performed with different levels of energy. Then the same test was performed for another method, which omits the metric for choosing clusterhead and forming clusters. Therefore, the clusterheads were chosen randomly and the clusters were shaped without any defined situation. The base station chooses the clusterheads based on the time that it receives the information. Then, the rest of the nodes join the clusters, without any specific consideration.

<table>
<thead>
<tr>
<th>Network connectivity Proposed algorithm</th>
<th>Network connectivity Random algorithm</th>
<th>Network average remaining battery</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.6056</td>
<td>0.5691</td>
<td>0.90</td>
</tr>
<tr>
<td>0.6126</td>
<td>0.5765</td>
<td>0.90</td>
</tr>
<tr>
<td>0.6172</td>
<td>0.5672</td>
<td>0.90</td>
</tr>
<tr>
<td>0.6105</td>
<td>0.5752</td>
<td>0.90</td>
</tr>
<tr>
<td>0.6085</td>
<td>0.5909</td>
<td>0.90</td>
</tr>
<tr>
<td>0.6050</td>
<td>0.5809</td>
<td>0.90</td>
</tr>
<tr>
<td>0.6042</td>
<td>0.5433</td>
<td>0.88</td>
</tr>
<tr>
<td>0.6027</td>
<td>0.5558</td>
<td>0.88</td>
</tr>
<tr>
<td>0.6062</td>
<td>0.5311</td>
<td>0.88</td>
</tr>
<tr>
<td>0.6034</td>
<td>0.5274</td>
<td>0.88</td>
</tr>
<tr>
<td>0.6048</td>
<td>0.5231</td>
<td>0.88</td>
</tr>
<tr>
<td>0.5952</td>
<td>0.5102</td>
<td>0.88</td>
</tr>
<tr>
<td>0.5940</td>
<td>0.5423</td>
<td>0.85</td>
</tr>
<tr>
<td>0.5909</td>
<td>0.5456</td>
<td>0.85</td>
</tr>
<tr>
<td>0.5806</td>
<td>0.5473</td>
<td>0.85</td>
</tr>
<tr>
<td>0.5927</td>
<td>0.5454</td>
<td>0.85</td>
</tr>
<tr>
<td>0.5953</td>
<td>0.5471</td>
<td>0.85</td>
</tr>
<tr>
<td>0.5941</td>
<td>0.5519</td>
<td>0.85</td>
</tr>
<tr>
<td>0.5752</td>
<td>0.5372</td>
<td>0.83</td>
</tr>
<tr>
<td>0.5865</td>
<td>0.5056</td>
<td>0.83</td>
</tr>
<tr>
<td>0.5786</td>
<td>0.5438</td>
<td>0.83</td>
</tr>
<tr>
<td>0.5675</td>
<td>0.5264</td>
<td>0.83</td>
</tr>
<tr>
<td>0.5662</td>
<td>0.5344</td>
<td>0.83</td>
</tr>
<tr>
<td>0.5727</td>
<td>0.5607</td>
<td>0.83</td>
</tr>
</tbody>
</table>
Table 4 - first test result

The result of these series of tests indicates that in a stable situation and for fixed structure, the network connectivity of the proposed algorithm is always better and higher than the random algorithm.

In addition, another difference between the proposed algorithm and the random algorithm, is that on a same situation (nodes battery, nodes placement, same set of nodes), the proposed algorithm choose the same clusterheads and it forms the same clusters. However, each time the random algorithm chooses a different clusterhead and forms different clusters. In all the pictures below, the red node represents the base station, the yellow node is a clusterhead that at least has one node in its cluster, the green node is an empty clusterhead and blue nodes are simple nodes. (Nodes distances in these pictures are short only to show the different colours and roles of the nodes, and the effect of using the link metric on the network topology.)

Figure 19 – Established network, proposed algorithm, first and second run
To complete the tests and in order to check the effect of the mobility on the network connectivity, another set of tests were performed. Four different situations were considered and each time the network connectivity was reported.

First a simple node was moved far from the rest of the nodes. As the results showed in Table 5, the network connectivity decreased.

In these series of tests, the results indicates that even when a node moves far, the network connectivity of the proposed algorithm is still better and higher that the random algorithm.

For the second test, an empty clusterhead was moved. In these tests, the network connectivity was reported for both algorithms and the result reported in Table 6 shows that although the network connectivity decreased, the proposed algorithm still has a better connectivity.
Test of Solution

<table>
<thead>
<tr>
<th>Proposed algorithm</th>
<th>Random algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connectivity before node movement</td>
<td>Connectivity after node movement</td>
</tr>
<tr>
<td>0.5431</td>
<td>0.5350</td>
</tr>
<tr>
<td>0.5467</td>
<td>0.5376</td>
</tr>
<tr>
<td>0.5473</td>
<td>0.5330</td>
</tr>
<tr>
<td>0.5489</td>
<td>0.5360</td>
</tr>
<tr>
<td>0.5490</td>
<td>0.5346</td>
</tr>
</tbody>
</table>

Table 5 - Test result - a simple node moved far

<table>
<thead>
<tr>
<th>Proposed algorithm</th>
<th>Random algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connectivity before node movement</td>
<td>Connectivity after empty clusterhead movement</td>
</tr>
<tr>
<td>0.5431</td>
<td>0.5001</td>
</tr>
<tr>
<td>0.5467</td>
<td>0.5046</td>
</tr>
<tr>
<td>0.5473</td>
<td>0.5019</td>
</tr>
<tr>
<td>0.5489</td>
<td>0.5007</td>
</tr>
<tr>
<td>0.5490</td>
<td>0.5014</td>
</tr>
</tbody>
</table>

Table 6 - Test result - an empty clusterhead moved far

For the third test, a clusterhead that had nodes in its cluster was moved. Also, this time the network connectivity was reported for both algorithms. The result reported in Table 7 shows that although the network connectivity decreased, the proposed algorithm still has a better connectivity.

<table>
<thead>
<tr>
<th>Proposed algorithm</th>
<th>Random algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connectivity before node movement</td>
<td>Connectivity after clusterhead movement</td>
</tr>
<tr>
<td>0.5431</td>
<td>0.4251</td>
</tr>
<tr>
<td>0.5467</td>
<td>0.4386</td>
</tr>
<tr>
<td>0.5473</td>
<td>0.4279</td>
</tr>
<tr>
<td>0.5489</td>
<td>0.4319</td>
</tr>
<tr>
<td>0.5490</td>
<td>0.4275</td>
</tr>
</tbody>
</table>

Table 7 - Test result - a clusterhead moved far
In order to have an accurate estimation of the effect of the mobility on the network connectivity, in another test, a simple node was moved to two different distances and network connectivity was reported for both algorithms. The result presented in table 8 for this test indicates that the network connectivity of the proposed algorithms is better and higher than the random algorithm.

<table>
<thead>
<tr>
<th>Proposed algorithm</th>
<th>Random algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network Connectivity</td>
<td>Distance 1 Connectivity</td>
</tr>
<tr>
<td>0.6008</td>
<td>0.5869</td>
</tr>
<tr>
<td>0.6012</td>
<td>0.5811</td>
</tr>
<tr>
<td>0.6053</td>
<td>0.5872</td>
</tr>
<tr>
<td>0.6048</td>
<td>0.5762</td>
</tr>
<tr>
<td>0.6044</td>
<td>0.5740</td>
</tr>
</tbody>
</table>

Table 8 - Test result - One node moved to two different distances
6 Results of Test, Measurement or Simulation

After several times establishing the network, first based on the proposed algorithm and then based on the random algorithm (same situation), it was seen that the proposed method always chooses the same clusterheads and forms the same clusters, while the random variation acts differently each time.

The most important result is that, the proposed algorithm provides a more reliable and stable network, while for random algorithm network, one node was always about to lose connectivity (blinking) and once the network lost one node.

Considering the result of average network connectivity, the proposed algorithm is more connected, while it is more reliable. The results are summarized in Figures 22 and 23.

![Figure 22 - Network comparison](image1)

![Figure 23 - Connectivity comparison between proposed algorithm and random algorithm](image2)
Wireless Sensor Network Group Connectivity

As the results show, the proposed algorithm forms a more reliable network, while it has a better connectivity. Over a period of losing energy, the proposed algorithm has less fluctuation. Furthermore, the network does not lose any node, while in the random algorithm, always one node is blinking, which indicates that it is about losing connectivity.

As a comparison for different scenarios that had been deployed, the average of the network connectivity was calculated and shown as a graph in Figure 24.

![Effect of mobility on network connectivity](image)

Figure 24 - Effect of mobility on network connectivity

In addition, Figure 25 presents a graph that shows the average network connectivity for a node movement to two different distances. This graph also shows that the proposed algorithm has a better connectivity compared to the random algorithm, even when a node goes very far.

![Effect of different distances on network connectivity](image)

Figure 25 - Effect of two different distance of connectivity

Figures 24 and 25 indicate that, the proposed algorithm has always a better connectivity if compared to the random one. Based on the results, it is possible to conclude that using the link metric to form the network indeed can help to improve the connectivity.
7 Conclusions and Suggestions to Future Work

Within the experiments performed in this thesis work, it was concluded that a mobile WSN can have the connectivity improved and the network is more reliable and stable, if its deployment is based on link metric.

To have a well connected mobile WSN there are steps that can be followed. First the topology of the network should be considered. This topology may arrange the nodes in a way that it reduces the unnecessary communications and balances energy consumption, while the network stays well connected and reliable. The topology may arrange the nodes in different levels and connect them based on a predefined condition which is calculated by a link metric formula. The network deployment has to be done in such a way to distribute the network monitoring among all the nodes. The first advantage of this approach is that it prevents a single node to loose its energy due to the fact that it is monitoring the whole network. Therefore energy consumption is distributed among all the nodes. The second advantage is that since the network is established based on the proposed metric, it stays connected longer and presents to be more reliable during its lifetime. In addition, the network can reconfigure itself only when the worst case happens. This way, unnecessary rearrangement of the nodes is prevented. Therefore, the network is able to save energy due to optimization of the communication used to keep connectivity.

Future work

An important study that can be done as a constitution of this thesis is to improve the connectivity formula, so that it shows more precisely the network connectivity fluctuations for different scenarios.

In order to improve the conductivity measure and have a more realistic network connectivity value, weakest links of each cluster can be studied more precisely. The result can then be used to improve the connectivity measure. One suggestion is using the minimum link quality of each cluster to measure the connectivity.

Another important issue that can be addressed is considering other factors and includes them in the metric. One of these factors is hop count and the other can be nodes direction.

After the experiment, it was noticed that network latency is one of the aspects that can be studied and be improved. To improve this experiment, the effect of different sets of K value for metric formula on network latency can be studied. Then a best formula can be chosen considering connectivity and latency. It is more accurate and efficient to choose the metric that has a reliable connectivity and stays stable during the network lifetime while it decreases network latency.

Considering that nodes are mobile and they may be used in different environments, effects of different environmental situations, like direction, distance and obstacles in between can be studied.

Another aspect that can be studied is comparing the cost of cluster or network reconfiguration when a clusterhead goes far from the network, and then compares it with the cost of the current experiment method.

Another suggestion to improve the experiments performed in this thesis work is to choose a maximum and minimum number for clusterheads, and reconfigure the network based on these maximum and minimum numbers of clusterheads. In the current experiments, the network will be reconfigured when it has only one clusterhead.
Conclusions and Suggestions to Future Work

References


Wireless Sensor Network Group Connectivity


[27] Rodrigo Fonseca, Omprakash Gnawali, Kyle Jamieson, Philip Levis. UC Berkeley, Univ. of Southern California, MIT CSAIL, Stanford Univ, "Four-Bit Wireless Link Estimation".

[28] Alec Woo and David Culler, "Evaluation of Efficient Link Reliability Estimators for Low-Power Wireless Networks".

[29] Qing Ling, Zhi Tian, "Impact of mobility on topology control of wireless sensor networks", Department of Electrical and Computer Engineering, Michigan Technological University Houghton, Michigan, USA.

Conclusions and Suggestions to Future Work

Science and Engineering The University of Texas at Arlington, Arlington, TX 76019, USA.


[33] "Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications for Low Rate Wireless Personal Area Networks (LR-WPANs)", IEEE std. 802.15.4 – 2003.

[34] 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver, ChipCon products, Texas instrument, 2008.


[39] Indranil Saha, Lokesh Kumar Sambasivan, Ranjeet Kumar Patro, Subhas Kumar Ghosh, " Distributed Fault-Tolerant Topology Control in Static and Mobile Wireless Sensor Networks", Honeywell Technology Solutions Lab Pvt. Ltd. 151/1, Doraisanipalya, Bannerghatta Road, Bangalore 560 076, India.


[41] Olivier Doussel, Patrick Thiran and Martin Hasler, "Connectivity in ad-hoc and hybrid networks", LCA,LANOS-ISC-FI&C (Inst. of Communication Systems), EPFL: Swiss Federal Institute of Technology, CH-1015 Lausanne, Switzerland
Wireless Sensor Network Group Connectivity
Conclusions and Suggestions to Future Work

8 Appendix

Sunspot Java Code

//StartApplication.java

package org.sunspotworld;

import com.sun.spot.peripheral.Spot;
import com.sun.spot.sensorboard.EDemoBoard;
import com.sun.spot.sensorboard.peripheral.ISwitch;
import com.sun.spot.sensorboard.peripheral.ITriColorLED;
import com.sun.spot.peripheral.radio.RadioFactory;
import com.sun.spot.peripheral.radio.IRadioPolicyManager;
import com.sun.spot.io.j2me.radiostream.*;
import com.sun.spot.io.j2me.radiogram.*;
import com.sun.spot.util.*;
import java.io.*;
import javax.microedition.io.*;
import javax.microedition.midlet.MIDlet;
import javax.microedition.midlet.MIDletStateChangeException;

/**
 * The startApp method of this class is called by the VM to start the
 * application.
 * *
 * The manifest specifies this class as MIDlet-1, which means it will
 * be selected for execution.
 * */
public class StartApplication extends MIDlet {

    private ITricolorLED [] leds = EDemoBoard.getInstance().getLEDs();

    protected void startApp() throws MIDletStateChangeException {
        System.out.println("Hello, world");
        new BootloaderListener().start();  // monitor the USB (if
        connected) and recognize commands from host

        long ourAddr =
        RadioFactory.getRadioPolicyManager().getIEEEAddress();

        System.out.println("Our radio address = "+
        IEEEAddress.toDottedHex(ourAddr));

        ISwitch sw1 =
        EDemoBoard.getInstance().getSwitches()[EDemoBoard.SW1];

        leds[0].setRGB(100,0,0);          // set color to moderate
        red

        while (sw1.isOpen()) {            // done when switch is
        pressed
            leds[0].setOn();            // Blink LED
            Utils.sleep(250);        // wait 1/4 seconds
            leds[0].setOff();
            Utils.sleep(1000);      // wait 1 second
        }

        notifyDestroyed();          // cause the MIDlet to exit
    }

    protected void pauseApp() {
        // This is not currently called by the Squawk VM
    }
}
import com.sun.spot.peripheral.Spot;
import com.sun.spot.sensorboard.EDemoBoard;
import com.sun.spot.sensorboard.peripheral.ISwitch;
import com.sun.spot.sensorboard.peripheral.ITriColorLED;
import com.sun.spot.sensorboard.peripheral.radio.RadioFactory;
import com.sun.spot.sensorboard.peripheral.radio.IRadioPolicyManager;
import com.sun.spot.io.j2meRadiogram.*;
import com.sun.spot.io.j2meRadiostream.*;
import com.sun.spot.util.*;
import java.io.*;
import javax.microedition.io.*;
import javax.microedition.midlet.MIDlet;
import javax.microedition.midlet.MIDletStateChangeException;
import java.util.*;
import java.lang.*;

public class BaseStation extends MIDlet implements ISwitchListener {

    private ITriColorLED[] leds = EDemoBoard.getInstance().getLEDs();
    private ISwitch[] switches = EDemoBoard.getInstance().getSwitches();

    //this is the vector for keeping the informations of the cluster heads
    Vector clusterheads = new Vector();

    //this vectore is to keep the information for the connectivity of the
    clusters
    Vector Connectivity = new Vector();

    //we will be using this object to save the informations in the vector
    NodeInfo SimpleNodeInfo = new NodeInfo();

    //we will be using these objects to arrange the elements of the vectors
    //as well as reading them for sending messages to them
    NodeInfo back = new NodeInfo();
Conclusions and Suggestions to Future Work

NodeInfo Ivalue = new NodeInfo();
NodeInfo Jvalue = new NodeInfo();
NodeInfo CH = new NodeInfo();
NodeInfo CHupdate = new NodeInfo();

// we specify the node degree to be 1 which means that this is the base station
int Ndspecification = 1;

// for receiving messages from other nodes
// private RadiogramConnection ReceiveConn = null;
// private Radiogram Receivedg;

// for sending messages to other nodes
private RadiogramConnection SendConn = null;
private Radiogram Senddg;

// the received message from a specific node
String address = null;
String Nodeadd = null;
String CHaddress = null;

// the value of the metric after calculating it
double ReceivedMetric = 0;

// these are the weights that we will be using in the link metric
double BK1 = 0.60;
double BK2 = 0.30;
double BK3 = 0.1;

protected void startApp() throws MIDletStateChangeException {
    System.out.println("Communication MIDlet");
    System.out.println("my specification is " + Ndspecification);

    switches[0].addISwitchListener(this);
    switches[1].addISwitchListener(this);

    try {
        RadiogramConnection ReceiveConn = (RadiogramConnection)Connector.open("radiogram://:123");
        Radiogram Receivedg = (Radiogram)ReceiveConn.newDatagram(200);
    }
}
while (true) {
    try {
        ReceiveConn.receive(Receivedg);
        NodeInfo SimpleNode = new NodeInfo();
        Message = Receivedg.readUTF();
        SimpleNode.NodeMetric = Receivedg.readDouble();
        SimpleNode.NodeDegree = Receivedg.readInt();
        SimpleNode.NodeAddress = Receivedg.getAddress();

        // the received RSSI
        double signal = Receivedg.getRssi();
        // System.out.println("the received RSSI is " + signal);
        // the received LQI
        double quality = Receivedg.getLinkQuality();
        // System.out.println("the received lqi is " + quality);

        // System.out.println("received = = = = " + SimpleNode.NodeMetric);
        int bat = Spot.getInstance().getPowerController().getBattery().getBatteryLevel();
        double batdouble = (double) bat;

        double linksignal = ((signal + 30)*100)/(60*100);
        double lqi = (quality * 100)/(255*100);
        // LQI is between 0 and 255 and we receive -LQI
        double linkbattery = (batdouble/100);
        // System.out.print("my batter is " + linkbattery);
        ReceivedMetric = MetricCalc(linkbattery, linksignal, lqi);
        // System.out.println("the calculated received metric is " + ReceivedMetric);

        switch (SimpleNode.NodeDegree) {
            case 4:
                if (!Receivedg.isBroadcast() && Message.equals("im new")) {
                    // System.out.println("the recieved address is " + SimpleNode.NodeAddress);
                    // System.out.println("A simple node is recieved ");
                    // System.out.println("the recieved value is " + SimpleNode.NodeMetric);
                    clusterheads.addElement(SimpleNode);
                    // System.out.println("the elements of vector: " + clusterheads);
                } else {
                    //
                }
                break;
        }
    }
}
Conclusions and Suggestions to Future Work

// the base station is receiving from a clusterhead
\n\ncase 2:
\n    if (Receivedg.isBroadcast()){
\n    }
\n    break;
\n    }

if (Message.equals("my new uptodate metric")){
\n    int n = Connectivity.size();
    for (int i = 0 ; i < n ; i++){
        CHupdate = (NodeInfo)
        Connectivity.elementAt(i);
        if
        (CHupdate.NodeAddress.equals(Receivedg.getAddress())){
        Connectivity.setElementAt(SimpleNode, i);
        }
    }
\n    for (int i = 0 ; i < n ; i++){
        CHupdate = (NodeInfo)
        Connectivity.elementAt(i);
\n    }else if (Message.equals("my battery")) { \n\n        double CHmetric = MetricCalc(SimpleNode.NodeMetric,
        linksignal, lqi);
\n        double averagemetric =
        ((CHmetric+ReceivedMetric)/2);
\n        if (CHmetric < 0.50){
            NumOfComeBack++;
            if (NumOfComeBack == 3 ){
                NumOfComeBack = 0;
            }
            }else{
\n            }
\n\n    int n = Connectivity.size();
    for (int i = 0 ; i < n ; i ++){
        CHupdate = (NodeInfo)
        Connectivity.elementAt(i);
        if
        (CHupdate.NodeAddress.equals(Receivedg.getAddress())){
            SimpleNode.NodeMetric =
            ((CHupdate.NodeMetric+averagemetric)/2) ;
\n        Connectivity.setElementAt(SimpleNode, i);
Wireless Sensor Network Group Connectivity

```java
for (int i = 0 ; i < n ; i ++){
    CHupdate = (NodeInfo) Connectivity.elementAt(i);
}

int Chcounter = Connectivity.size() ;
double totalMetric = 0;
int NumofNodes = 0;

for (int i = 0 ; i< Chcounter ; i++){
    CH = (NodeInfo) Connectivity.elementAt(i);
    totalMetric = totalMetric +
    CH.NodeMetric;
    NumofNodes = NumofNodes +
    CH.NodeDegree;
}

    totalMetric = totalMetric/Chcounter;
    double FinalConn = totalMetric * ((NumofNodes) * (NumofNodes - 1))/((NetworkNodes) * (NetworkNodes - 1));

    System.out.println("the network connectivity is "+ FinalConn);
}

else if (Message.equals("ClusterHead")){
    SimpleNode.NodeMetric = (SimpleNode.NodeMetric +
    ReceivedMetric)/2;
    Connectivity.addElement(SimpleNode);
    for (int i = 0; i <8; i++) {
    leds[i].setRGB(255, 0, 0);
    leds[i].setOn();
    }
}else if (Message.equals("searching")){
    int n = Connectivity.size();
    int i = 0;
    while (i < Connectivity.size()) {
        CH = (NodeInfo) Connectivity.elementAt(i);
        if (CH.NodeAddress.equals(Receivedg.getAddress())){
            Connectivity.removeElementAt(i);
        }
        i++;
    }
}
```

```java
} catch (IOException ex) {
    System.out.println("Error receiving packet: " + ex);
    ex.printStackTrace();
}
```
Conclusions and Suggestions to Future Work

```java
public void Transmite(String msg, String add, double link, int deg) {
    try {
        SendConn = (RadiogramConnection)Connector.open("radiogram://" +
            add + ":123");
        Senddg = (Radiogram)SendConn.newDatagram(200);
        Senddg.reset();
        Senddg.writeUTF(msg);
        Senddg.writeDouble(link);
        Senddg.writeInt(deg);
        SendConn.send(Senddg);
        SendConn.close();
    } catch (IOException ex) {
        System.out.println("Error opening connections: " + ex);
        ex.printStackTrace();
    }
}

public double MetricCalc(double Benergy, double rssi, double lqi) {
    metric = (BK1 * Benergy + BK2 * rssi + BK3 * lqi);
    return metric;
}

protected void pauseApp() {
}

protected void destroyApp(boolean unconditional) throws MIDletStateChangeException {
}

public void switchPressed(ISwitch sw) {
    double linkmetric=0;
    double x=0;
    double connectivity=0;
    int NumOfCH = 2;
    if (sw == switches[0]) {
        // broadcasting the address
        try {
            SendConn = (RadiogramConnection)Connector.open("radiogram://broadcast:123");
            Senddg = (Radiogram)SendConn.newDatagram(200);
            Senddg.reset();
            Senddg.writeUTF(address);
        }
    }
```
Senddg.writeDouble(1);
Senddg.writeInt(1);

SendConn.send(Senddg);
SendConn.close();

for (int i = 0; i < 8; i++) {
    leds[i].setRGB(255, 0, 255);
    leds[i].setOn();
}

} catch (IOException ex) {
    System.out.println("Error sending packet: "+ ex);
    ex.printStackTrace();
}

//the base staion will wait for three seconds to have received everything
//then it will chose the cluster heads
Utils.sleep(4000);

int n = clusterheads.size();

System.out.println("the number of the elements in the link is "+ n);

//here we want to arrange the objects in the vectore from the one
//with the maximmume value for the node link so the bast station
can pick the best to be cluster heads

for (int i = 0; i < n-1; i++) {
    for (int j = i + 1; j < n; j++) {
        Ivalue = (NodeInfo) clusterheads.elementAt(i);
        Jvalue = (NodeInfo) clusterheads.elementAt(j);
        if (Ivalue.NodeMetric < Jvalue.NodeMetric) {
            clusterheads.setElementAt(Jvalue, i);
            clusterheads.setElementAt(Ivalue, j);
        }
    }
}

//here we read the values of the vectoe
for (int k = 0; k < n; k++) {
    back = (NodeInfo) clusterheads.elementAt(k);
    x = back.NodeMetric;

    //here the base station is chosing the best valuse of the link
    //metric
    //and will use it to chose the cluster heads
    for (int l = 0; l < NumOfCH; l++) {
        CH = (NodeInfo) clusterheads.elementAt(l);
        CHaddress = CH.NodeAddress;

        System.out.println("the address of the " + l + " cluster
head is " + CHaddress);
        Transmit("*** you are a cluster head ***", CHaddress, 1,
Ndsspecification);
Conclusions and Suggestions to Future Work

}  
//we only wanna save the informations of the clusterheads in the vectore  
//so now we will delete the informations of the remaining nodes in the vecotre  
//System.out.println("total number of cluster head before removing is " + clusterheads.size());  
while (clusterheads.size()>NumOfCH){  
    int i = NumOfCH;  
    clusterheads.removeElementAt(i);  
    //System.out.println("the node " + i + " had been removed");  
}  
//System.out.println("total number of cluster head after removing is " + clusterheads.size());  
for (int i = 0; i < clusterheads.size() ; i ++){  
    back = (NodeInfo) clusterheads.elementAt(i);  
    //System.out.println(back.NodeAddress);  
}  
Utils.sleep(5000);  
int Chnumber = clusterheads.size() ;  
// Start from the first CH to store nodes information on simplenodes vector  
for (int i = 0 ; i< Chnumber ; i++){  
    CH = (NodeInfo) clusterheads.elementAt(i);  
    CHaddress = CH.NodeAddress ;  
    Transmit("**Connectivity**", CHaddress, 1, Ndspecification);  
}  
Utils.sleep(2000);  
//here when we press the right bottom of the base station  
//it will send a msg to the clusterheads asking them to send their connectivity informations  
int Chcounter = Connectivity.size() ;  
// Start from the first CH to store nodes information on simplenodes vector  
double totalMetric = 0;  
int NumofNodes = 0;  
for (int i = 0 ; i< Chcounter ; i++){  
    CH = (NodeInfo) Connectivity.elementAt(i);  
    //System.out.println("recieved number of nodes is " + CH.NodeDegree + " from " + CH.NodeAddress );  
    //System.out.println("recieved link metric is " + CH.NodeMetric + " from " + CH.NodeAddress );  
    totalMetric = totalMetric + CH.NodeMetric;  
    NumofNodes = NumofNodes + CH.NodeDegree;  
}
public void switchReleased(ISwitch sw) {
}

//NodeInfo.java
	package org.sunspotworld;

	/**
	* this is a class that we will use object from to save in the vectore
	* each object consist of three parameters
	* the node address, the link metric of a node and the degree of the node
	* the degree of the node is 1 if it is the basestation
	* 2 if it is a cluster head, and 3 if it is a regular node
	*/

	public class NodeInfo {
		public String NodeAddress;
		public double NodeMetric;
		public int NodeDegree;

}
Conclusions and Suggestions to Future Work

// Nodes.java

package org.sunspotworld;

import com.sun.spot.peripheral.Spot;
import com.sun.spot.sensorboard.EDemoBoard;
import com.sun.spot.sensorboard.peripheral.ISwitch;
import com.sun.spot.sensorboard.peripheral.ITriColorLED;
import com.sun.spot.sensorboard.peripheral.radio.RadioFactory;
import com.sun.spot.sensorboard.peripheral.radio.IRadioPolicyManager;
import com.sun.spot.io.j2me.radiostream. *
import com.sun.spot.io.j2me.radiogram. *
import com.sun.spot.sensorboard.peripheral.ISwitchListener;
import com.sun.spot.util. *
import java.io. *
import javax.microedition.io. *
import javax.microedition.midlet.MIDlet;
import javax.microedition.midlet.MIDletStateException;
import java.lang. *
import java.util. Random;
import java.util. *
import java.lang. Math;
import java.util. Timer;
import java.util. TimerTask;

/**
 * this is the application that will run on all the nodes in the network except the base station
 * according to this application the node will become either a clusterhead or will remain as a simple node
 */

public class Nodes extends MIDlet implements ISwitchListener {

    private ITriColorLED [] leds = EDemoBoard.getInstance().getLEDs();
    private ISwitch switches[] = EDemoBoard.getInstance().getSwitches();
    int val=100;
    int j=0;
    int cmd=0;

    String SimplenodeAddress = null ;
    double SimplenodeMetric = 0 ;
    int Simplenodespecification = 0;
    double recieved;

    //the Ndspecification is the current degree of the node (the one that is receiving)
    int Ndspecification = 4 ;
    int NDdegree = 0;

    String address=null;

    private RadiogramConnection tx = null;
    private Radiogram xdg;
    String add="hi";
    double linkbattery;

    Vector links = new Vector();
    Vector NodeLinks = new Vector();
NodeInfo Ivalue = new NodeInfo();
NodeInfo Jvalue = new NodeInfo();
NodeInfo compare = new NodeInfo();
NodeInfo CH = new NodeInfo();
NodeInfo CNode = new NodeInfo();

double TransmitMetric;
double ReceivedMetric;

boolean exist = false;

double link = 0;
int Nodecounter = 0;
int bat;

double clustermetric = 0;
double TotalCHMetric = 0;

String NodesInCluster = null;

double NK1 = 0.30;
double NK2 = 0.60;
double NK3 = 0.1;
int NumOfComeBack = 0;

String ComeBack = null;
//to keep the address of the base station in the cluster heads for
//future uses
String BaseStation = null;

//for if i node came to the network and for 5 minutes it didn't recive
//anything from other nodes
Timer timer = new Timer();
Timer SendTimer = new Timer();
Timer NewCH = new Timer();
Timer CHSendTImer = new Timer();

String Clustersize = "empty";

String message = null;

protected void startApp() throws MIDletStateChangeException {
    System.out.println("Communication MIDlet");

    switches[0].addISwitchListener(this);

    long addressValue =
    Spot.getInstance().getRadioPolicyManager().getIEEEAddress();
    address=IEEEAddress.toDottedHex(addressValue);

    try {
        //open a connection to recieve the messages
        RadiogramConnection rx =
        (RadiogramConnection)Connector.open("radiogram://:123");
Conclusions and Suggestions to Future Work

Radiogram rdg = (Radiogram) rx.newDatagram(200);

    // the timer will repeat the task every 5 seconds after the
    first 10 seconds delay
    timer.schedule(new NewNode(), 10000, 5000);

    while (true) {
        try {
            // receiving the packets and putting them in the object
            // nodes of the class NodeInfo
            rx.receive(rdg);

            NodeInfo nodes = new NodeInfo();

            message = rdg.readUTF();
            recievied = rdg.readDouble();
            nodes.NodeDegree = rdg.readInt();
            nodes.NodeAddress = rdg.getAddress();

            // the received RSSI
            double signal = rdg.getRssi();
            System.out.println("the received RSSI is " + signal);

            // the received LQI
            double quality = rdg.getLinkQuality();
            System.out.println("the received LQI is " + quality);

            // The remaining battery
            bat = Spot.getInstance().getPowerController().getBattery().getBatteryLevel();

            double batdouble = (double) bat;
            System.out.println("received ===== " + recievied);

            double linksignal = ((signal + 30) * 100) / (60 * 100);
            double lqi = (quality * 100) / (255 * 100); // LQI is between 0 and 255 and we receive -LQI
            linkbattery = (batdouble / 100);

            TransmitMetric = MetricCalc(linkbattery, linksignal, lqi);

            ReceivedMetric = MetricCalc(recievied, linksignal, lqi);

            System.out.println("THE RECEIVED METRIC IS " + ReceivedMetric);

            // the metric of the node is based on the received signal
            // and its own battery
            nodes.NodeMetric = TransmitMetric;

            switch (Ndspecification) {
                // the node is a clusterhead
                case 2:
                // the node is a clusterhead receiving from a cluster head
                if (nodes.NodeDegree == 2) {
//if the node is a cluster head and it has received a msg from a regular node
//it will know that this node is in the cluster of the cluster head
else if (nodes.NodeDegree == 3 &&
message.equals("im new") && !message.equals("my battery") &&
!message.equals("i did") && !message.equals("searching")){

    nodes.NodeMetric = received;
    links.addElement(nodes);

    int z = links.size();

    clustersize = "not empty";

    for (int i = 0; i < z; i ++){
        Ivalue = (NodeInfo)
        System.out.println(Ivalue.NodeAddress +
" is in my cluster");
        System.out.println("the link metric of node "+ Ivalue.NodeAddress + " is "+ Ivalue.NodeMetric);
    }
    for (int i = 0; i < 8; i++){
        leds[i].setRGB(255,255,0);
        leds[i].setOn();
    }
}

//we want to calculate the connectvity of our network
//so the cluster head will send to its nodes to get their connectivity
else if (nodes.NodeDegree == 1 &&
message.equals("**Connectivity**" )){

    System.out.println("Sending metric and number of nodes to basestaion");
    // Get the number of the nodes in the cluster
    // we will get them from the vector that has the information of the nodes
    Nodecounter = links.size();

    //if the node is a cluster head that doesn't have any nodes connected to it
    if (Nodecounter == 0 ){
        Transmit("ClusterHead",
        rdg.getAddress(), TransmitMetric, 1);
        System.out.println("my
        metric is "+ TransmitMetric);
    //System.out.println("Nodeth information has been sent to " +
    rdg.getAddress()); 
    }

    // if the node is a cluster head and it has other nodes in its cluster
    // it has to send the connectivity of all the nodes in its cluster
    else {
        // Start from the first Node to store nodes information on simplenodes vector
        for (int i = 0; i <
        Nodecounter; i++){
Conclusions and Suggestions to Future Work

CH = (NodeInfo) links.elementAt(i);
// SimplenodeAddress = CH.NodeAddress;
SimplenodeMetric = SimplenodeMetric + CH.NodeMetric;
// Simplenodespecification = Ndspecification;
System.out.println("the node " + CH.NodeAddress + " has the metric of " + CH.NodeMetric);
}
TotalCHMetric = SimplenodeMetric / Nodecounter;
System.out.println("the total cluster metric is "+ TotalCHMetric);
Transmit("ClusterHead", rdg.getAddress(),
TotalCHMetric, Nodecounter+1);
//System.out.println("Nodeth information has been sent to " +
rdg.getAddress());
TotalCHMetric = 0;

}

//when the network is established the nodes should keep sending their informations every two seconds
//either to the cluster head or if it is a cluster head to the base station
else if (nodes.NodeDegree == 1 &&
message.equals("**Network established**")){
System.out.println("network is established we will send to the nodes now");
// if the cluster head is not connected to any node it will only send it's own metric
// but we have to keep repeating that
Nodecounter = links.size();
if (Nodecounter == 0 ){
Nodecounter = 1;
TotalCHMetric = TransmitMetric;
// here we will only send the remaining battery and the rest will be calculated at the base station
CHSendTimer.schedule(new SendInfotoBS(), 1000, 5000);
// System.out.println("my battery is " + TransmitMetric);

//the Clusterhead will start to send it's own information to the asestation
}else{
// if the Ch has nodes in its cluster is should farward the messgae to them
// and then they should keep sending their informations to the cluster head
// and the cluster head will pass the connectivity along with its own link to the base station
}
else{
// if the Ch has nodes in its cluster is should farward the messgae to them
// and then they should keep sending their informations to the cluster head
// and the cluster head will pass the connectivity along with its own link to the base station
for (int i = 0 ; i< Nodecounter ; i++){
Wireless Sensor Network Group Connectivity

```
CNode = (NodeInfo) links.elementAt(i);
Transmit("**Network established**", CNode.NodeAddress, 0, Ndspecification);
System.out.println("information about the establishemnet sent to " + CNode.NodeAddress);
}
}
else if (nodes.NodeDegree == 1 && message.equals("find a new connection")){
Transmit("searching", rdg.getAddress(), TransmitMetric, Ndspecification);
Ndspecification = 4;
for (int i = 0; i < 8; i ++){
leds[i].setRGB(0, 0, 0);
leds[i].setOn();
}
int n = links.size();
for (int j = 0; j < n; j ++){
CNode = (NodeInfo) links.elementAt(j);
NodeLinks.addElement(CNode);
}
links.removeAllElements();
NewCH.schedule(new RepeatedNode(), 1000, 5000);
}
//now the network is established and the nodes will keep sending there informations to the blusterheads
else if (nodes.NodeDegree ==3 && message.equals("my battery")){
System.out.println("__________________________________________");
System.out.println("the new battery received from the node "+ rdg.getAddress()+ " is " + recieved);
System.out.println("the new metric received from the node "+ rdg.getAddress()+ " is " + ReceivedMetric);
System.out.println("__________________________________________");
if (ReceivedMetric<0.50){
System.out.println("asking the node " + rdg.getAddress() + " to come back ");
Transmit("Come Back", rdg.getAddress(), 1, Ndspecification);
NumOfComeBack++;
if (NumOfComeBack == 3 ){
//if the node didn't come back after sending the notification for 5 times
//the Cluster head will ask it to reset itself and search for a new cluster head
//and it will delete it from its link
```

80
Conclusions and Suggestions to Future Work

Transmit("find a new CH", rdg.getAddress(), 1, Ndspecificaiton);
    NumOfComeBack = 0;

} else{
}
} else {

//we are replaceing the nodes information with the new informations that we got
//to keep our vectore updated with the latest numbers
    int n = links.size();
    for (int i = 0 ; i < n ; i ++){
        CNode = (NodeInfo)links.elementAt(i);
        if (CNode.NodeAddress.equals(rdg.getAddress())){
            nodes.NodeMetric = ReceivedMetric;
            links.setElementAt(nodes, i);
        }
    }

    //we are printing out the new informations in the vector
    for (int i = 0 ; i < n ; i ++){
        CNode = (NodeInfo)links.elementAt(i);
        System.out.println("the new element "+ i + " in the metric is " +
            CNode.NodeMetric );
    }
    SimplenodeMetric = 0;
//we are calculating the new metric of the cluster to send it to the base station
    Nodecounter = links.size();
    for (int i = 0 ; i< Nodecounter ; i++){
        CH = (NodeInfo)links.elementAt(i);
        SimplenodeMetric = SimplenodeMetric + CH.NodeMetric ;
        System.out.println("the node " + CH.NodeAddress + " has the metric of " +
            CH.NodeMetric);
    }
    TotalCHMetric = SimplenodeMetric / Nodecounter ;
    System.out.println("the total cluster metric is " + TotalCHMetric);
    Transmit("my new uptodate metric", BaseStation, TotalCHMetric,
        links.size()+1);
    bat = Spot.getInstance().getPowerController().getBattery().getBatteryLevel();
    batdouble = (double) bat;
    linkbattery = (batdouble/100);
    Transmit("my battery", BaseStation, linkbattery, 1);
    TotalCHMetric = 0;
}

//the cluster head is far away and we want it to come back
else if(nodes.NodeDegree == 1 &&
    message.equals("Come Back")){

81
Wireless Sensor Network Group Connectivity

```java
for (int j = 0 ; j < 5 ; j++) {
    for (int i = 0 ; i < 8 ; i++){
        leds[i].setOff();
    }
    Utils.sleep(200);
    for (int i = 0 ; i < 8 ; i++){
        leds[i].setOn();
    }
    Utils.sleep(200);
}
```

// a new node has sent a broadcast that it wants to join the network
// the clusterhead will send its information to the node
// and the node will decide which cluster it wants to join

```java
else if (nodes.NodeDegree == 4 && rdg.isBroadcast()){
    System.out.println(" the node " + rdg.getAddress() + " is asking to join the network");
    Transmit("my info", rdg.getAddress(), TransmitMetric, Ndspecification);
    Utils.sleep(3000);
    Transmit("new node chose CH", rdg.getAddress(), TransmitMetric, Ndspecification);
}
```

// now the clusterhead will ask the new node to start sending its informations

```java
else if (nodes.NodeDegree == 3 && message.equals("i did")){
    Transmit("**Network established**", rdg.getAddress(), TransmitMetric, Ndspecification);
    for (int i = 0 ; i < 8 ; i++){
        leds[i].setRGB(255, 255, 0);
        leds[i].setOn();
    }
    //the cluster head will add the new node to its vector
    nodes.NodeMetric = recieved;
    links.addElement(nodes);
    Clustersize = "not empty";
    System.out.println("information about the establishement sent to " + CNode.NodeAddress) ;
}
```

else if (nodes.NodeDegree == 3 && message.equals("searching")){
    int n = links.size();
    int k = 0 ;
    while (k < links.size()){ 
        CNode= (NodeInfo)
    links.elementAt(k);
```
Conclusions and Suggestions to Future Work

```java
if (CNode.NodeAddress.equals(rdg.getAddress())){
    links.removeElementAt(k);
    k++;
}
Utils.sleep(1000);
if (links.isEmpty()){
    for (int j = 0; j < 8; j++){
        leds[j].setRGB(0, 255, 0);
        leds[j].setOn();
    }
    Clustersize = "empty";
    CHSendTImer.schedule(new SendInfoBS(), 1000, 5000);
}
break;
```

```java
  case 4:
  //the node is a regular node but it will be elected as a clusterhead
  if (nodes.NodeDegree == 1 && message.equals("*** you are a cluster head ***")){
    BaseStation = rdg.getAddress();
    // if the node received that it should be a cluster head
    //it will make its degree as 2 (cluster head)
    //then it will broadcast a msg to all the nodes around it sending its
    //address
    Ndspecification = 2 ;
    System.out.println("I am a
clusterhead");
    Utils.sleep(1000);
    for (int i = 0; i < 8; i++){
        leds[i].setRGB(0, 255, 0);
        leds[i].setOn();
    }
    try {
        tx = (RadiogramConnection)Connector.open("radiogram://broadcast:123");
        xdg = (Radiogram)tx.newDatagram(200);
        //send the addres of the node
        //when we press the left switch button
        xdg.reset();
        xdg.writeUTF(address);
        xdg.writeDouble(0);
        //the nodes will decided to which cluster to join based on their wn metric
        xdg.writeInt(Ndspecification);
        tx.send(xdg);
        tx.close();
    } catch (IOException ex) {
        System.out.println("Error
sending packet: " + ex);
```
//it will wait for 2 second so all the nodes has received the address
//then it will broadcast another msg asking the nodes to chose their
cluster head

Utils.sleep(2000);
try {

        tx = (RadiogramConnection) Connector.open("radiogram://broadcast:123");
        xdg = (Radiogram) tx.newDatagram(200);
        //send the addres of the node
        //when we press the left switch button

        xdg.reset();
        xdg.writeUTF("chose your(clusterhead)");
        xdg.writeDouble(0);
        xdg.writeInt(Ndspecification);
        tx.send(xdg);
        tx.close();
    } catch (IOException ex) {
        System.out.println("Error sending packet: "+ ex);
        ex.printStackTrace();
    }
}

//this is the first step in the network, the node is a simple node
//receiving a broadcast to send the address to the base station to elect
the cluster head

else if (nodes.NodeDegree == 1 && rdg.isBroadcast()){

    System.out.println(" **** the received address is **** "+ nodes.NodeAddress);
    Transmit("im new", rdg.getAddress(), TransmitMetric, Ndspecification);
    System.out.println("the link metric of this spot is "+ TransmitMetric);
    System.out.println("i am sending to "+ rdg.getAddress());
}

//now the node will receive a message from the clusterhead and it will
decide that it is a simple node

else if (nodes.NodeDegree == 2 && !message.equals("chose your clusterhead") && !message.equals("**Network established**") && !message.equals("Come Back")){

    Ndspecification = 3;
    links.addElement(nodes);
Conclusions and Suggestions to Future Work

System.out.println("I am a regular node");
int n = links.size();

for (int i= 0; i <n; i ++){
    Ivalue = (NodeInfo) links.elementAt(i);
    System.out.println("the item "+
i + " in the link is " + Ivalue.NodeMetric);
    System.out.println("the received value of the "+ i + "metric is " +Ivalue.NodeMetric);
}

for (int i = 0; i < n-1 ; i++ ){
    for (int j = i+1; j < n ; j++ ){
        Ivalue = (NodeInfo) links.elementAt(i);
        Jvalue = (NodeInfo) links.elementAt(j);
        System.out.println("Ivalue = "+ Ivalue.NodeMetric);
        System.out.println("Jvalue = "+ Jvalue.NodeMetric);
        if ( Ivalue.NodeMetric < Jvalue.NodeMetric){
            links.setElementAt(Jvalue, i);
            links.setElementAt(Ivalue, j);
        }
    }
}

CH = (NodeInfo) links.elementAt(0);

// if the node is receiving infromatin from a simple node and it wants to chose among it for its metric
else if (nodes.NodeDegree == 3 && !message.equals("i did") && !message.equals("searching") && !message.equals("chose your clusterhead")){
    Ndspecification = 3;
    links.addElement(nodes);

    System.out.println("I am a regular node");
    int n = links.size();

    for (int i= 0; i <n; i ++){
        Ivalue = (NodeInfo) links.elementAt(i);
        System.out.println("the item "+
i + " in the link is " + Ivalue.NodeMetric);
        System.out.println("the received value of the "+ i + "metric is " +Ivalue.NodeMetric);
    }
}
for (int i = 0; i < n-1; i++) {
    for (int j = i+1; j < n; j++) {
        Ivalue = (NodeInfo) links.elementAt(i);
        Jvalue = (NodeInfo) links.elementAt(j);
        System.out.println("Ivalue = " + Ivalue.NodeMetric);
        System.out.println("Jvalue = " + Jvalue.NodeMetric);
        if (Ivalue.NodeMetric < Jvalue.NodeMetric) {
            links.setElementAt(Jvalue, i);
            links.setElementAt(Ivalue, j);
        }
    }

    CH = (NodeInfo) links.elementAt(0);
}

break;

case 3:

    // the node is a simple node and it was asked to choose its cluster head in
    if (nodes.NodeDegree == 2 && message.equals("chose your clusterhead") && rdg.getAddress().equals(CH.NodeAddress)) {
        CH = (NodeInfo) links.elementAt(0);
        System.out.println("the cluster head is " + CH.NodeAddress);
        System.out.println("the clusterhead metric is " + CH.NodeMetric);
        for (int i = 0; i < 8; i++) {
            leds[i].setRGB(0, 0, 255);
            leds[i].setOn();
        }

        Transmit("im new", CH.NodeAddress, CH.NodeMetric, Ndspecification);
    }

    else if (nodes.NodeDegree == 3 && message.equals("new node chose CH") && rdg.getAddress().equals(CH.NodeAddress)) {
        CH = (NodeInfo) links.elementAt(0);
        System.out.println("the cluster head is " + CH.NodeAddress);
Conclusions and Suggestions to Future Work

```java
System.out.println("the clusterhead metric is " + CH.NodeMetric);

for (int i = 0; i < 8; i++) {
    leds[i].setRGB(0, 0, 255);
    leds[i].setOn();
}

Transmit("i did", CH.NodeAddress, CH.NodeMetric, Ndspecification);

// the new node has connected to the network
else if (nodes.NodeDegree == 2 && message.equals("new node chose CH") && rdg.getAddress().equals(CH.NodeAddress)) {
    CH = (NodeInfo) links.elementAt(0);
    System.out.println("the cluster head is " + CH.NodeAddress);
    System.out.println("the clusterhead metric is " + CH.NodeMetric);
    for (int i = 0; i < 8; i++) {
        leds[i].setRGB(0, 0, 255);
        leds[i].setOn();
    }
    Transmit("i did", CH.NodeAddress, CH.NodeMetric, Ndspecification);
}

// if the node is receiving another message to chose its clusterhead it will drop it so the link wont get more informations
else if (nodes.NodeDegree == 2 && message.equals("chose your clusterhead") && !rdg.getAddress().equals(CH.NodeAddress)) {
}

else if (nodes.NodeDegree == 3 && message.equals("chose your clusterhead") && !rdg.getAddress().equals(CH.NodeAddress)) {
}

else if (nodes.NodeDegree == 2 && message.equals("**Network established**")) {
    //System.out.println("the network is established");
    //System.out.println("i will start sending my informations now");
    CNode = (NodeInfo) links.elementAt(0);
    Nodecounter = NodeLinks.size();
```

87
if (Nodecounter == 0) {
    System.out.println("the node metric "+ TransmitMetric);
    CHSendTImer.schedule(new SendInfo(), 1000, 5000);
}

//the Clusterhead will start to send it's own information to the basestation

} else{
    // if the CH has nodes in its cluster is should farward the message to them
    // and then they should keep sending their informations to the cluster head
    // and the cluster head will pass the connectivity along with its own link to the base station
    for (int i = 0 ; i< Nodecounter ; i++){
        CNode = (NodeInfo) NodeLinks.elementAt(i);
        Transmit("**Network established**", CNode.NodeAddress, CNode.NodeMetric, Ndspecification);
        System.out.println("information about the establishment sent to " + CNode.NodeAddress) ;
    }
}

//if the node is far away and it should come back
else if (nodes.NodeDegree == 2 && message.equals("Come Back")){
    for (int j = 0 ; j < 5 ; j++) {
        for (int i = 0 ; i < 8 ; i++){
            leds[i].setOff();
        }
        Utils.sleep(200);
        for (int i = 0 ; i < 8 ; i++){
            leds[i].setOn();
        }
        Utils.sleep(200);
    }
}

else if (nodes.NodeDegree == 3 && message.equals("Come Back")){
    for (int j = 0 ; j < 5 ; j++) {
        for (int i = 0 ; i < 8 ; i++){
            leds[i].setOff();
        }
        Utils.sleep(200);
        for (int i = 0 ; i < 8 ; i++){
            leds[i].setOn();
        }
        Utils.sleep(200);
    }
}

// now the node has to reset itself so it can start to find a new cluster head
else if (nodes.NodeDegree == 2 && message.equals("find a new CH")){

}
Conclusions and Suggestions to Future Work

Ndspecification = 4;

// sendThread.msg.equals("find a new CH");
links.removeAllElements();
for (int i = 0 ; i <8 ; i++){
    leds[i].setOff();
}

Transmit("searching", rdg.getAddress(),
TransmitMetric, 3);
NewCH.schedule(new RepeatedNode(),
1000, 5000);
}

else if (nodes.NodeDegree == 3 && message.equals("find a new CH")){
    Ndspecification = 4;
    //sendThread.msg.equals("find a new CH");
    links.removeAllElements();
    for (int i = 0 ; i <8 ; i++){
        leds[i].setOff();
    }
    Transmit("searching", rdg.getAddress(),
    TransmitMetric, 3);
    NewCH.schedule(new RepeatedNode(),
    1000, 5000);
}

else if (nodes.NodeDegree == 3 && message.equals("i did")){

    Transmit("**Network established***", rdg.getAddress(),
    TransmitMetric, Ndspecification);
    for (int i = 0 ; i <8 ; i ++){
        leds[i].setRGB(50, 50, 50);    
        leds[i].setOn();
    }
    //the cluster head will add the new node to its vector
    nodes.NodeMetric = recieved;
    NodeLinks.addElement(nodes);
    Clustersize = "not empty";
    System.out.println("information about the establishment sent to " + CNode.NodeAddress);
}

else if (nodes.NodeDegree == 4 &&
rdg.isBroadcast()){ System.out.println(" the node " +
rdg.getAddress() + " is asking to join the network ");
Transmit("my info",
rdg.getAddress(), TransmitMetric, Ndspecification);
    Utils.sleep(3000);
    Transmit("new node chose CH",
    rdg.getAddress(), TransmitMetric, Ndspecification);
}
else if { nodes.NodeDegree == 3 && message.equals("**Network established**") ){
    System.out.println("network is established we will send to the nodes now");
    // if the cluster head is not connected to any node it will only send it's own metric
    // but we have to keep repeating that
    Nodecounter = NodeLinks.size();
    if (Nodecounter == 0 ){
        CHSendTImer.schedule(new SendInfo(), 1000, 5000);
        // the Clusterhead will start to send it's own information to the basestation
    }else{
        // if the Ch has nodes in its cluster is should farward the messgae to them
        // and then they should keep sending their informations to the cluster head
        // and the cluster head will pass the connectivity along with its own link to the base station
        for (int i = 0 ; i< Nodecounter ; i++){
            CNode = (NodeInfo) NodeLinks.elementAt(i);
            Transmit("**Network established**", CNode.NodeAddress, CNode.NodeDegree, Ndspecification);
            System.out.println("information about the establishemnet sent to "+ CNode.NodeAddress);
        }
    }
}
else if {nodes.NodeDegree ==3 && message.equals("my battery")}){
    System.out.println("__________________________________________");
    System.out.println("the new battery received from the node "+ rdg.getAddress()+ " is "+ ReceivedMetric);
    System.out.println("__________________________________________");
    //
    if (ReceivedMetric < 0.10){
        if (ReceivedMetric < 0.50){
            System.out.println("asking the node "+ rdg.getAddress() + " to come back ");
            Transmit("Come Back", rdg.getAddress(), TransmitMetric, Ndspecification);
            NumOfComeBack++;
Conclusions and Suggestions to Future Work

if (NumOfComeBack == 3) {
    // if the node didn't come back after sending the notification for 5 times
    // the Cluster head will ask it to reset itself and search for a new cluster head
    // and it will delete it from its link
    Transmit("find a new CH",
             rdg.getAddress(), 1, Ndspecification);
    NumOfComeBack = 0;
} else {
    NumOfComeBack = 0;
}

// we are replacing the nodes information with the new informations that we got
// to keep our vector updated with the latest numbers
int n = NodeLinks.size();
System.out.println("the elements in the link now are " + n);

    for (int i = 0 ; i < n ; i ++){
        CNode = (NodeInfo) NodeLinks.elementAt(i);
        if (CNode.NodeAddress.equals(rdg.getAddress())){
            CNode.NodeMetric = ReceivedMetric;
            NodeLinks.setElementAt(nodes, i);
        }
    }

    // we are printing out the new informations in the vector
    for (int i = 0 ; i < n ; i ++){
        CNode = (NodeInfo) NodeLinks.elementAt(i);
        System.out.println("the new element " + i + " in the metric is " + CNode.NodeMetric );
    }

    SimplenodeMetric = 0;
    // we are calculating the new metric of the cluster to send it to the base station
    Nodecounter = NodeLinks.size();
    for (int i = 0 ; i < Nodecounter ; i++){
        CH = (NodeInfo) NodeLinks.elementAt(i);
        SimplenodeMetric + CH.NodeMetric ;
    }
    TotalCHMetric = SimplenodeMetric / Nodecounter;
    System.out.println("the total cluster metric is " + TotalCHMetric);
    Transmit("my new up-to-date metric", rdg.getAddress(), TotalCHMetric, Nodecounter+1);
System.out.println("Nodeth information has been sent to " + rdg.getAddress());
TotalCHMetric = 0;
bat = Spot.getInstance().getPowerController().getBattery().getBatteryLevel();
batdouble = (double) bat;
linkbattery = (batdouble/100);
Transmit("my battery", rdg.getAddress(), linkbattery, 1);
}
else if (nodes.NodeDegree == 3 && message.equals("searching")){
    int n = NodeLinks.size();
    int k = 0 ;
    while (k < links.size()){ CNode= (NodeInfo) links.elementAt(k);
        if (CNode.NodeAddress.equals(rdg.getAddress())) {
            links.removeElementAt(k);
            k++;
        }
    }
    Util.sleep(1000);
    if (NodeLinks.isEmpty()){
        for (int j = 0 ; j <8 ; j++){
            leds[j].setRGB(0, 0, 255);
            leds[j].setOn();
        }
        Clustersize = "empty";
    }
    break;
}
}catch (IOException ex) {
    System.out.println("Error receiving packet: " + ex);
    ex.printStackTrace();
}
}
}catch (IOException ex) {
    System.out.println("Error opening connections: " + ex);
    ex.printStackTrace();
}

protected void pauseApp() {
}

protected void destroyApp(boolean unconditional) throws MIDletStateChangeException {
}

public void switchPressed(ISwitch sw) {
}

public void switchReleased(ISwitch sw) {
}
Conclusions and Suggestions to Future Work

//this will calculate the metric of the network
public double MetricCalc (double Benergy, double rssi, double lqi){
    clustermetric = (NK1 * Benergy + NK2 * rssi + NK3 * lqi);
    return clustermetric;
}

//this is for a new node that wants to join the network
class NewNode extends TimerTask {
    public void run() {
        if (Ndspecification == 4){
            // the node is new and it will broadcast its information to the
            // cluster head
            // it will get answers from more than one and it will chose
            // to join the best one
            System.out.println("i am a new node to the network");
            try {
                tx = (RadiogramConnection)Connector.open("radiogram://broadcast:123");
                xdg = (Radiogram)tx.newDatagram(200);
                //send the address of the node
                // when we press the left switch button
                xdg.reset();
                xdg.writeUTF("want to join");
                xdg.writeDouble(0);
                xdg.writeInt(Ndspecification);
                tx.send(xdg);
                tx.close();
            } catch (IOException ex) {
                System.out.println("Error sending packet: " + ex);
                Ndspecification = 4;
                links.removeAllElements();
                for (int i = 0 ; i <8 ; i++){
                    leds[i].setOff();
                }
                NewCH.schedule(new RepeatedNode(), 1000, 5000 );
            }
        }
        else {  
            System.out.println("i am already in the network ");
            this.cancel();
        }
    }
}

the timer for a new node that wants to join the network again
//so it will keep broadcasting to all the nodes
class RepeatedNode extends TimerTask {
    public void run() {
        if (Ndspecification == 4){
            // the node is new and it will broadcast its information to
            // the cluster head
        }
    }
}
// it will get answers from more than one and it will chose to join the best one
System.out.println("i am a new node to the network");

try {
    tx = (RadiogramConnection)Connector.open("radiogram://broadcast:123");
    xdg = (Radiogram)tx.newDatagram(200);
    // send the address of the node
    // when we press the left switch button
    xdg.reset();
    xdg.writeUTF("want to join");
    xdg.writeDouble(0);
    xdg.writeInt(Ndspecification);
    tx.send(xdg);
    tx.close();
} catch (IOException ex) {
    Ndspecification = 4;

    links.removeAllElements();
    for (int i = 0; i < 8; i++){
        leds[i].setOff();
    }
    NewCH.schedule(new RepeatedNode(), 1000, 5000);
}

} else {
    System.out.println("i am already in the network");
    this.cancel();
}

} //we will create a new timer to keep sending the informations of the node to the clusterhead
//until the node have to change its cluster head
class SendInfo extends TimerTask {
    public void run() {
        try {
            if (!message.equals("find a new CH")){
                // the node is new and it will broadcast its information to the cluster head
                // it will get answers from more than one and it will chose to join the best one
                try {
                    bat = Spot.getInstance().getPowerController().getBattery().getBatteryLevel();

                    double batdouble = (double) bat;
                    linkbattery = (batdouble/100);

                    CH = (NodeInfo) links.elementAt(0);

                    System.out.println("i am sending my battery to " + CH.NodeAddress + " every 5 seconds " + linkbattery);
            
        } catch (Exception ex) {
            System.out.println("Error sending battery information");
        }
    
    } catch (Exception ex) {
        System.out.println("Error scheduling timer");
    }

    
    } //end of run method
}

} //end of class SendInfo


Conclusions and Suggestions to Future Work

```
import Radiogram

tx = (RadiogramConnection)Connector.open("radiogram://" + CH.NodeAddress + ":123");
xdg = (Radiogram)tx.newDatagram(200);
//send the addres of the node
//when we press the left switch button
xdg.reset();
xdg.writeUTF("my battery");
xdg.writeDouble(linkbattery);
xdg.writeInt(Ndspecification);
px.send(xdg);
px.close();
}
catch (IOException ex) {
   System.out.println("Error sending packet, i can not reach the destination");
   Ndspecification = 4;
   links.removeAllElements();
   for (int i = 0 ; i <8 ; i++){
       leds[i].setOff();
   }
   NewCH.schedule(new RepeatedNode(), 1000, 5000 );
}
else {
   System.out.println("i am joining a new CH ");
   this.cancel();
}
}
}

//if the clusterhead doesn't have any nodes connected
class SendInfotoBS extends TimerTask {
   public void run() {
      if (Clustersize.equals("empty")){
         // the node is new and it will broadcast its information to
         // the cluster head
         // it will get answers from more than one and it will chose
         // to join the best one
         try {
            bat = Spot.getInstance().getPowerController().getBattery().getBatteryLevel();
            double batdouble = (double) bat;
            linkbattery = (batdouble/100);
            System.out.println("i am sendinh my battery to the basestation every 5 seconds" + linkbattery);
         } catch (IllegalArgumentException exc){
            System.out.println("HIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
```
Wireless Sensor Network Group Connectivity

```java
public void Transmit(String msg, String add, double link, int deg) {
    try {
        try {
            tx = (RadiogramConnection) Connector.open("radiogram://" + add + ":123");
            xdg = (Radiogram) tx.newDatagram(200);
            // send the address of the node
            // when we press the left switch button
            xdg.reset();
            xdg.writeUTF("my battery");
            xdg.writeDouble(link);
            xdg.writeInt(deg);
            tx.send(xdg);
            tx.close();
        } catch (IOException ex) {
            System.out.println("Error sending packet: " + ex);
        }
    } catch (IllegalArgumentException exc) {
        System.out.println("BYEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE");
    }
}
```

Links removed: 
+ received packet qua Link metric

```java
// packet qua received Link metric
```