



Linköping University

INSTITUTE OF TECHNOLOGY

A Study on Smart Dust Networks

Maryam Abrishami

LITH-EX-11/0101

2011-05-31

Abstract

This thesis work is done for the department of Electronic System at The Institute of Technology at Linköping University (Linköpings Tekniska Högskolan). Study's focus is to design and implement a protocol for smart dust networks to improve the energy consumption algorithm for this kind of network.

Smart dust networks are in category of distributed sensor networks and power consumption is one of the key concerns for this type of network. This work shows that by focusing on improving the algorithmic behavior of power consumption in every network element (so called as mote), we can save a considerable amount of power for the whole network.

Suggested algorithm is examined using Erlang for one mote object and the whole idea has put into test for a small network using SystemC.

Acknowledgment

I would like to express my gratitude to my supervisor Dr. J. Jacob Wikner for introducing me to the topic as well for the support on the way, useful comments, remarks and engagement through the learning process of this master thesis. Furthermore I would like to thank Specifically Mr. Joe Armstrong and Mr. Jan Hederen for their patient and kind support through this thesis work. Also, I like to thank Mr. Fred Hebert, Mr. Parthaserathi Murali, Mr. Vahid Keshmiri and Mr. Joakim Alvbrant for their kind supports on the way.

List of abbreviations

Abbreviation		
WSN	Wireless Sensor Networks	A network of modules that are interconnected through a wireless communication interface
FHSS	Frequency Hopping Spread Spectrum	A kind of modulation that different version of IEEE 802.11 covers different frequency bandwidths for this modulation in indoor and outdoor applications
DSSS	Direct Sequence Spread Spectrum	A kind of modulation that different version of IEEE 802.11 covers different frequency bandwidths for this modulation in indoor and outdoor applications
SOM	self-organizing Map	
MEMS	Microelectromechanical Sensors	
MANET	Mobile Ad-hoc Network	
WMN	Wireless Mesh Network	
DARPA	Defense Advanced Research Projects Agency	
PRNET	Packet Radio Networks	
ALOHA	Areal Locations of Hazardous Atmospheres	
CSMA	Carrier Sense Medium Access	
SURAN	Survivable Adaptive Radio Networks	
GloMo	Global Mobile Information Systems	
NTDR	Near-term Digital Radio	
NTDRS	Near Term Digital Radio System	
IETF	Internet Engineering Task Force	
eNB	E-UTRAN (or Evolved) Node B	
SON	Self-Organizing Network	
D-SON	Distributed Self-Organizing Network	
C-SON	Centralized Self-Organizing Network	

A Study on Smart Dust Networks	1
Maryam Abrishami	1
LITH-EX-11/0101	1
2011-05-31	1
1 Introduction	10
1.1 Motivation	10
1.2 Working environment	10
1.3 Work specification	11
1.4 Outline of thesis	11
2 Distributed networks	13
2.1 Introduction	13
2.2 Wireless Sensor Networks (WSNs)	13
2.3 Smart Dust	14
2.3.1 What is a smart dust network?	15
2.3.2 Examples of existing applications smart dust	15
2.3.3 Future applications for smart dust	15
2.4 Other network types?	16
2.4.1 Ad-hoc Network	16
2.4.1.1 Ad-hoc Sensor Networks.....	17
2.4.1.2 Ad-hoc Mobile Networks.....	18
2.4.2 Ad-hoc Standards	18
2.4.3 Autonomous Sensor Networks	18
3 Artificial Intelligence for Smart Dust Networks	20
3.1 Introduction	20
3.2 Artificial Intelligence	20
3.2.1 Fuzzy logic	20
3.3 self-organizing Networks (SON)	21
3.3.1 Introduction	21
3.3.2 SON Architectural types	21
3.3.2.1 Distributed SON.....	22
3.3.2.2 Centralized SON.....	22
3.3.2.3 Hybrid SON	22
3.3.3 SON Functions	22
3.3.3.1 Self-configuration.....	23
3.3.3.2 Self-optimization.....	23
3.3.3.3 Self-healing.....	23
3.3.3.4 SON Self-Configuration.....	23
3.3.3.5 SON Self-Optimization.....	24
3.3.3.5.1 Self-optimizing motivation.....	24
3.3.3.5.2 Self-optimizing network functionality.....	25
3.3.3.6 SON Self-Healing.....	27
3.3.3.6.1 Self-healing network basics.....	27
3.3.3.6.2 Cell Degradation	27
3.3.3.6.3 Cell outage compensation.....	28
4 Smart Dust Networks	29
4.1 Introduction	29
4.2 Opportunities and challenges	29
4.3 Smart Dust Mote Architecture	30
4.4 Power management strategies in Smart Dust	31

4.4.1	Harvard Event-driven Architecture	31
4.4.2	Low power design	31
4.5	Sensors used on dust motes	32
4.6	Communication methods in Smart Dust	32
4.6.1	Radio Frequency transmission	33
4.6.2	Optical communication techniques	33
4.6.2.1	Optical communication – Main category: Free-Space.....	34
4.6.2.1.1	Optical communication – Sub-category of Free-Space: Active and Passive.....	34
4.6.2.2	Fiber-optic communication.....	34
4.7	Routing Protocols	35
4.7.1	Single Hop Model	35
4.7.2	Multi-hop Model	36
4.7.3	Cluster-based Hierarchical Model	36
4.7.3.1	Uniform Clustering.....	37
4.7.3.2	non-Uniform Clustering.....	38
4.8	Network topologies	39
4.8.1	Dynamic networks	39
4.8.1.1	Basic Definitions in Dynamic Network Protocol.....	39
4.8.1.2	Data propagation Algorithm in Dynamic Network Protocol.....	40
4.8.1.3	Advantages.....	40
4.8.1.4	Disadvantages.....	40
4.8.2	Semi Dynamic networks	40
4.8.3	Static networks	40
4.8.3.1	Advantages.....	41
4.8.3.2	Disadvantages.....	41
4.9	Data Propagation algorithms in Smart Dust	42
4.9.1	Geographical distance issues	42
4.9.2	Jamming	42
5	Study on simulators for Smart Dust networks	43
5.1	Introduction	43
5.1	Simulators for Sensor Networks	43
5.1.1	GloMoSim	43
5.1.2	System C	43
5.1.3	Tiny OS	43
5.1.4	NS-2	43
5.1.5	SensorSim	44
5.1.6	JavaSim	44
5.1.7	Erlang	44
5.1.8	OPNET	44
5.1.9	Other simulators	44
5.2	Erlang	44
5.2.1	What is Erlang?	45
5.2.2	Erlang History	45
5.2.3	Erlang Actor model	45
5.2.4	Erlang download, installation and other	45
5.2.5	Erlang Applications	45
5.2.6	Concurrency in Erlang	46
5.2.7	Distributed Programming in Erlang	46
5.2.7.1	Distributed Erlang.....	47
5.2.7.2	Socket-based distribution.....	47
6	Algorithms and Policies	48
6.1	Introduction	48
6.2	A general view on our algorithm in Erlang	48
6.3	Power management policy	48
6.3.1	Mote behavior algorithm	49
6.3.1.1	Sleeping Mode.....	50
6.3.1.2	Sensing Mode.....	50
6.3.1.3	Receiving Mode.....	51
6.3.1.4	Sending Mode.....	51

6.3.1.4.1	Sending after Sensing.....	51
6.3.1.4.2	Sending after Receiving.....	51
6.3.1.5	Harvesting Mode.....	51
6.3.2	An example of energy usage by motes	51
6.3.3	Policies in case of possible clashing	53
7	Erlang Simulation Results	54
7.1	Introduction	54
7.2	Mote.cc functionality	54
7.3	mote_main.cc functionality	55
7.4	mote_gen.cc functionality	55
8	Conclusions	56
8.1	Future work	57
8.2	Suggestions	57
9	References	58
10	Appendices	61
10.1	Appendix 1: Erlang Codes	61
10.1.1	mote_bhv	61
10.1.2	base_station	63
10.1.3	mote_event	64
10.1.4	base_station_protocol	66
10.1.5	user_protocol	67
10.2	Appendix 2: C-Codes	68
10.2.1	Mote.cc	68
10.2.2	mote.h	77
10.2.3	mote_gen.h	78
10.2.4	out.txt	79

List of Figures

Figure 1: A general view of a distributed sensor network.....	14
Figure 2: The centralized and distributed architecture type of SON network.....	22
Figure 3: A review of SON functions.....	23
Figure 4: Different key factors of SON functions.....	28
Figure 5: A smart dust mote, containing different parts such as sensors and other network equipment. [35].....	30
Figure 6: A simple view of single hop communication routing model.....	36
Figure 7: A simple view of multi-hop transmission in wireless sensor network.....	36
Figure 8: A view of cluster based routing model.....	37
Figure 9: A 3D view of an uniform clustering with two decision levels.....	37
Figure 10: From left to right, A view of: A- A non-clustering network; B- A uniform clustering with two levels of information fusion; C- A uniform clustering with three levels of information fusion.....	38
Figure 11: An example of non-uniform clustering with total number of 20 sensors.....	38
Figure 12: A simple presentation of Dynamic networks and the way of ID assignment	39
Figure 13: A simple view of a Static Network; Fixed place for every mote and base station can be seen clearly.....	41
Figure 14: A presentation of distance measurement in a static network.....	41
Figure 15: A simple view of the system and direction of data flow.....	48
Figure 16: A simple presentation of different energy usage levels for a mote	49
Figure 17: The FSM that describing mote's behavior.....	50
Figure 18: An example which shows the change of energy levels in a mote for different modes.....	52

List of Tables

Table 1: Category of Sensors types [40].....32

1 INTRODUCTION

This thesis introduces a new way for decreasing the power consumption for Smart dust networks, an efficient approach to monitoring the behavior of each network member and improve their power consumption's algorithm in order to manage this very important feature of smart dust networks and overcome the obstacle of power management in this type of networks.

Communication networks in today's world are serving more and more number of purposes and individuals which results in increasing importance of all types communication networks. Invention and deployment of new technological trends is a response to an increasing demand for advanced network structures such as distributed sensor networks. Self-organizing networks and smart dust networks as part of distributed sensor network category, has an important role in well-defined use cases.

A proper network management policy which focuses on improvement the behavior of smart dust networks and resolving the existed problems in this type of networks, is of the subject of many research works. One of the problems which has a high degree of importance and is an interesting subject of research is increasing and maintaining the life time of smart dust networks. Power consumption, management of the related behavior, monitoring and analysis and control of network behaviors to ensure lower power consumption is the fundamental of our work presented in this thesis.

In the main studies which have had major effect on designing smart dust networks, with the purpose of increasing the life time of this kind of networks, focus is mainly on improving the network's behavior as a whole. For example, Pister and Kahn introduced a new concept in which the basis was to minimizing the power consumption due to transmission as much as possible. This concept is discussed in more details, later on, in this thesis work.

Intelligent and automatic monitoring of each network node, called mote, is the conception brought forward in this thesis to address the power consumption and therefore, smart dust network lifetime problems. The focus of this study is to design and implement a protocol for smart dust networks to improve the energy consumption for this type of network. The major difference in this study is basically introducing a new way which concentrate on improving the algorithmic behavior of power consumption in every mote. This way, we will show that , this concept will achieve a lower power consumption for smart dust network. Suggested algorithm is implemented using Erlang for one mote object while this concept, as a whole, has put into test for a small network, using SystemC.

1.1 Motivation

Smart dust networks are in category of distributed sensor networks and power consumption is one of the key concerns for this type of network. This thesis work Study's focus is to design and implement a protocol for smart dust networks to improve the energy consumption for this type of network.

This research tries by focusing on improving the algorithmic behavior of power consumption in every mote, achieve a lower power consumption for smart dust network. Suggested algorithm is examined using Erlang for one mote object and the whole idea has put into test for a small network using SystemC.

1.2 Working environment

The study and research of this project was carried out in the Electronic System division of Linköping University. The working environment was good and all the required facilities were provided.

Smart Dust project is a big project consisted of Dr. Jacob Wikner as the main supervisor and 10 master students. All the research were done as master thesis of those students. Different areas of the smart dust project was given to each/pair of students to work on. There were students working on digital signal processing, logic circuits, Analog-to-digital converters, energy harvesters, regulator and networks. A couple of the PhD students in the Electronics System division were also assisting the students in guidance and finding solutions.

The group had one weekly meeting each week in which everyone would discuss the work that was done during the week, feedback, comments, questions and future plans. We also had brainstorming to help solve a particular issue. Hence everyone would have a knowledge of the function or progress of the other members topics.

All the group members (students) were international master students. This created a good cultural mix in order to have a well functioning working group. We had students from India, Pakistan, Iran and Egypt.

1.3 Work specification

This thesis work, introduces a new protocol in smart-dust network for decreasing the power consumption of the network. The concept is examined based on an example network, which simulates a self-organizing network with primary purpose of detecting fire in large jungles. The supposed jungle is remote and not easily accessible by human, so motes will be thrown down into the area, they will self-organize themselves and form a self-organized wireless sensor network.

Every mote is designed to have different energy states based on the situation of the environment. The introduced concept has been implemented in Erlang programming language to show and simulate the expected behavior of each mote. Most of message control between motes in the network is assumed to be handled by the base station. In this work, we will mainly focus on data types, link types between different processes and a general view of the system.

Using Erlang we introduced and implemented a new algorithm which focuses on moving the mote to the most energy efficient state, based on the network situation and event-detecting concept that is implemented using SystemC. The motes implemented using Erlang, are autonomous and are capable to self-organize.

This study also introduces and implements an algorithm using SystemC, in which we will be able to add a fixed number of motes. Using SystemC, a network of motes is implemented which simulate a smart dust network in which every mote, behaves based on the protocol we have introduced in this thesis work. Also using SystemC, we have implemented an event-detecting and event-driven system, in which the concept is not about passing information through the network, instead it is about detecting and alarming a goal event (like fire in the jungle in our example case) to a base station. As soon as, alarm rises inside that network, base station takes all the required actions to protect the network and fulfill the purpose of the network.

Using Erlang and SystemC, we were able to examine the new protocol in a network with fixed predefined number of motes. The code and therefore, the algorithm can be extended to handle the alternative in which we will be able to add an infinite number of motes. This concept can be improved so adding the number of motes can be done initially and the system is able to handle to add more motes once the system is already up and running.

1.4 Outline of thesis

Chapter 1 motivates this thesis work and talk about the outline of this thesis work as well as work specification.

Chapter 2, focuses on distributed networks as the mother of Smart Dust network and wireless sensor networks. This chapter introduces Smart Dust networks, talking about the history of their existence and shows some implemented examples using this technology to motivate the reader.

Chapter 3, is an introduction to artificial technologies and the possibility of using them in Smart Dust networks. This chapter also has a look on self-organizing Networks and their characteristics.

Chapter 4, can very well be the longest chapter of this document and the technical basis for this thesis work. This chapter introduces different technical subject regarding Smart Dust Networks like different topologies, mote structures and routing protocols.

Chapter 5, looks specifically to all possible simulators that can be used to implement the characteristics of a Smart Dust network. Most of the studied software and simulators, have been used in other research works in different universities around the world. Introducing all these simulators, then we introduce our choice of simulator and explain later in the same chapter which characteristics of this simulator makes it perfect for our work.

Chapter 6, is the heart of this research work and the result of the pre-studies and introductions in previous chapters. This chapter introduces the algorithm that has been designed in this thesis work to solve the power consumption issues in Smart Dust network. The algorithm is introduced and is explained in detail using figures and sketches to make it more interesting and easier to understand for the reader.

Chapter 7, has a brief introduction to the Erlang implementation. Later in this chapter, every function is explained in detail using figures and examples. Codes are attached in appendices part at the end of this document and can be studied.

Chapter 8 has a fast look on what has been done during this thesis work, weaknesses and advantages of the design. It is also suggested what can be done in future to improve this work and decrease the weaknesses. Other areas connected to the subject of this thesis are briefly introduced as well.

2 DISTRIBUTED NETWORKS

2.1 Introduction

In 1960 when Paul Baran, introduced his idea about distributed networks, no one could believe the incredible applications that he promised to be implementable using his idea. Applications such as online shopping which is now a normal process of everyone's daily life, looked so impossible at the time. Hence made AT&T to refuse investing on his unique ideas about distributed networks. Nine years later in 1969, when a network called Arpanet was introduced by Americans, based on Baran proposal, world started to believe that he was right. [17]

Distributed network is a term used for a network structure which its specific feature is distribution of the network resources (such as switching equipment and processors) on different geographical area in the sense of size, that these networks are working in.

Many examples of distributed networks can be seen around us, like the phone networks. Even there are some examples of natural distributed networks, such as a network of people who are working on the same subject but they are spread in a geographical area or a distributed network of animals.

However we should confess that Internet is the largest distributed network in its kind which is distributed computer network. Internet has grown incredibly fast from the time it has started to develop as DARPA net in 1960. Now Internet is a huge network which connects the world together. More than hundred countries in the world are now connected through Internet and more than a quarter of world population using it regularly in their daily life. [24]

Everything in the world is reshaping to fit to applications and services compatible with the internet. This can cover different areas; from the world of publications e.g. books and newspapers to the world of communication and information. It also creates a vast range of jobs, companies, university study fields, travel agencies and so on. Internet is still expanding with a fast rate and would find different shape and applications in near future. It is expected that a different and vast range of applications for internet is found in near future. An example can be "internet of things" which will connect everything in earth and space together, so that you can control many things, like your home appliances, even from space!

Distributed network is the basis for lots of different networks, in which each has the basic structure of distributed networks with some different features and special usage. One of the most interesting research areas which also belongs to this category of networks is distributed sensor networks that has a vast and interesting range of applications.

In this chapter we will discuss wireless sensor networks as a type of distributed networks and in continue we will address the "Smart Dust" network as a branch of distributed sensor networks which all are in category of ad-hoc networks. Then we will see what applications does smart dust have and what challenges researchers are facing in this research area. And at the end, we will try to consider other types of networks which are all related to our discussion about distributed networks.

2.2 Wireless Sensor Networks (WSNs)

Wireless Sensor Networks, as it can be understood by their name, are a collective number of sensors or sensing devices that are spread in an environment for monitoring purposes. These sensors are generally anonymous, aiming to track, detect and possibly report different phenomena in the environment they are distributed in. This phenomena can be temperature, humidity, pollution, etc., or even a combination of them.

This group of networks are a branch of ad-hoc network, in the category of distributed networks. Wireless sensor network is an interesting research area which makes the science world to hold lots of conferences and workshops in this area of research every year. It showed its importance to the world soon after it was born such that IEEE 802.11 was specifically designed to cover this important subject. The IEEE 802.11 is a set of standards which makes special rules for implementation of wireless sensor networks under the IEEE rules.

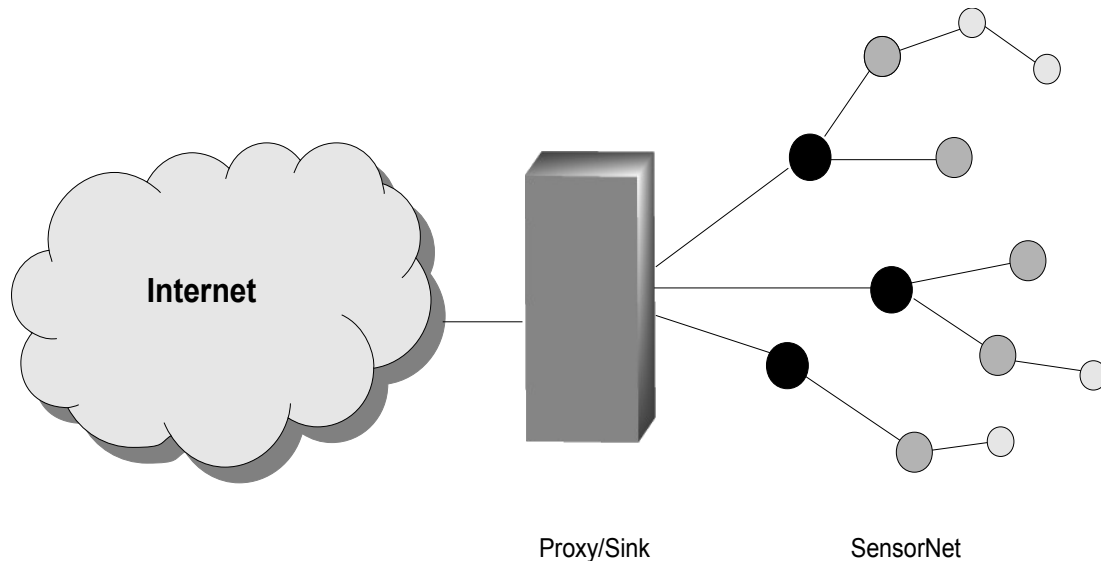


Figure 1: A general view of a distributed sensor network

The history of the IEEE for distributed networking goes back to 1985 and after that lots of other specification added to the standard and made it a big family of IEEE. Standards such as '802.11.a' addressing OFDM modulation, '802.11.b' which process DSSS modulation and '802.11.g' which considers both OFDM and DSSS modulation but for the different range of indoor and outdoor applications. This standard covers the implementation of the computer communication with the frequency bandwidth in the range of 2.4 and 3.6 and 5 Giga hertz. [22]

WSN has a large domain of applications. There are some successful research applications such as “Great Duck Island” by Berkeley in 2002, “Sniper detection” by Vanderbilt in 2003 and “Wireless blood pressure monitor” by Harvard in 2007 and much more as real applications or research areas. [18]

2.3 Smart Dust

Smart Dust belongs to the class of wireless sensor networks which together they make a special type of network belonging to the ad-hoc networks class. Smart dust networks and wireless sensor networks share some common characteristics and challenges as well as having differences.

Smart Dust basically points to large scale wireless sensor networks which contain a large number of low power and small (cubic millimeter until now) computational elements that are made by MEMS technology and are called motes. Motes automatically organize themselves in a network using the concept of self-organizing maps (SOM). This gives them the opportunity to function in the network autonomously. It means they will organize or possibly reorganize themselves in the network without human manipulation and maintenance. When a mote settles in the network, it starts a series of activities such as creating a connection with other motes. They function as small computational elements with especial behavior and specific task, provide their own power by harvesting energy from environment like solar energy, communicate with other motes using message passing with a selected wireless communication technology such as RF and so on. Smart Dust has a vast and unusual range of applications from detecting a possible fire in a huge forest to provide the security in a building.

More details about this type of networks which is our interest during this thesis work report, will be recalled again in chapter 4.

2.3.1 What is a smart dust network?

Smart Dust basically points to a large scale wireless sensor network which contain a large number of low power and very small (cubic millimeter until now) computational elements that are made by MEMS technology and are called motes.

When a mote settles in the network, it starts a series of activities such as creating a connection with other motes, functioning as a small computational element with especial behavior and specific responsibility, communicate with other motes by message passing using a selected wireless communication technology such as RF or optical communication.

Smart Dust has lots of interesting range of potential applications from detecting a possible fire in a huge forest to providing the security in a building.

2.3.2 Examples of existing applications smart dust

There are huge number of applications that can utilize Smart Dust technology. As the motes reach better performance and qualifications, and at the same time becoming smaller and consuming less power, a new range of potential applications can be applied as well. What a Mote is capable of performing, is a good measure how vast range of applications can be provided using this technology. [29]

There are a huge number of applications possible using this technology such as chemical, industrial and biological applications, business applications, quality control, tracking real time events, supervision of a large scale area like a national forest and many more. [29]

One of the early applications for Smart Dust was vehicle tracking in a desert located in a remote area. This system could detect any kind of movement in different areas. It can be used for example to detect enemies in a battlefield or observe animals, insects and plants behavior in biological applications. This project was implemented by U.S. Marine Corps and deployed for a real world movement detection purposes at Palm Spring desert in California. In this project motes could take benefit of self-organizing techniques to be placed in a network and perform the communications without human intervention. An airplane flew over the area and dropped eight motes randomly over Spring Palm area. Motes started to collect information about vehicle movements and sent them back successfully to the airplane. This information could be successfully retrieved to a computer for further analyzing. [29]

Another similar but in laboratory scale application from ETH University in Switzerland related to smart dust networks was a system for tracking the location of real-world phenomena. They used a sample car as a simulation of a real-world event. [28]

Sailor research group at University of California San Diego did a project on biological application where they tried to produce the motes of type chemical compounds. Therefore, it can be expected that such motes can be deployed for chemical supervisory purposes or for detecting different molecules or different chemical elements like gases existed in an environment. This application can be critically useful in terrorist attacks, in a battlefield or even in chemical plants. The real project still could not work in more than laboratory environment and it's not ready to be applied in an industrial scale yet. In the mentioned laboratory experiment, the produced chemical motes could detect hydrocarbon vapors. [29]

2.3.3 Future applications for smart dust

It's nice to have an idea about the future of smart dust. Berkeley Sensor and Actuator Center which has been funded by DARPA, has some future plans and possible applications for smart dust which make it possible to imagine how far we will be able to go using this kind of networking technology. They have already planned some military and also commercial application for smart dust and they have already started to work on. Below some interesting application which BSAC has predicted for smart dust network and started to work on them has been discussed.

One of the possible applications is called "Inventory control". The main idea here is a remote control for the home appliances. It allows the owner to be able to track and control different tools like truck, warehouses and so on remotely while you are away. So one can control the condition and state of each of his/her home appliances whenever and wherever he/she wants to. [23]

The other application for smart dust which can be expected to be implemented in near future is "Smart office spaces". We always have problems setting the air conditioner to the degree appropriate for everyone. This problem gets more serious when lots of people are at the same place like an office. Imagine that you will have some kind of network of sensors which are sewn to your cloths and continuously reports your body condition to a controller, and that in turn controls the air conditioner in your office. Getting the correct input information, air conditioner can run in the way which is proper for at least majority of people in an environment or possibly all of the people at the same place. So you and your office mates can enjoy the proper air condition without having any kind of complaints. [23]

The smart dust technology also can be employed to serve disable people and make their life much easier. One idea about this application comes with a mail from a disable man to Berkeley Sensors and Actuators Center. He suggested that we can design some kind of "quadriplegic mote" and we can somehow put it on a disable face. So this mote can detect and monitor specific face reactions and translate those facial reactions to some commands. These commands can be used to start and control different devices like computer and wheelchair or even a car. This can be a huge step for disables independence. [23]

One of the use case which can be a possibility to fabricate using this technology involves changing the keyboard definition as we have it today to a what which is so called "virtual keyboard". In this hypothesis, scientist think that it will be possible to replace the current keyboards to smart motes on our fingers which will translate our meaningful finger movements, to real action sin our computer. This concept, in general offers a whole new way of connecting to your computers, laptops and all the similar devices. This is a revolutionary concept which will make the life much easier for people with special needs. For example, based on this concept, we shall be able to play a piano without touching an actual physical piano and this shall open up for a whole new world which our imagination can have a free and active role in it.

2.4 Other network types?

There are several different field of studies in networking which focus on different kind of networks with different range of properties and specific range of applications. All these fields are so active and have very good potential for research. Here we shortly name some of them to attract reader's attention toward other kinds of interesting field of networking as well.

One of the most interesting topics in networking which also covers our topic is Ad-hoc networks which can be classified in more detailed in three different groups, consisting of "Mobile Ad-hoc Networks" (MANET), "Wireless Mesh Networks" (WMN), that will be discussed below briefly and of course "Wireless Sensor Networks" (WSN) which has been discussed before in 2.2.

2.4.1 Ad-hoc Network

Ad-hoc network also known as short-live network consists of two or more mobile devices that are connected together without any interface or any helping structure. If we compare ad-hoc network with the fixed wireless network, there are some interesting specification which may make ad-hoc network more advantageous in some cases and make it more specific in applications it can be employed. These characteristics makes an ad-hoc network to be able have a low cost in different aspects compare with fixed wireless network and also be rather easy in the sense of setting up and so on in areas which are in far geographical distances. The power of coverage for ad-hoc network can be increased if it can be integrated with a larger network. Increasing the coverage power will add lots of different applications for this kind of network. [26]

Ad-hoc networks can be categorized into three different generations. The current ad-hoc networks mostly belong to the third generation of these kind of networks. [25]

The origin of ad-hoc network (first generation) dates back to 1972 when Defense Advanced Research

Projects Agency (DARPA), invested a fund on packet switching radio communication research to see if is a feasible means of communication. Eventually it lead to formation of the first generation ad-hoc network, which mostly found its applications in military. DARPA PRNET (Packet Radio Networks) is also another product of the first generation ad-hoc networks which evolved between 1973 and 1987. [27] PRNET associated with ALOHA (Areal Locations of Hazardous Atmospheres) and CSMA (Carrier Sense Medium Access) resulted in medium access control, mostly used in combat environments. [25]

The second generation ad-hoc networks started to develop from 1980 and continued until 1993 and mostly was part of the SURAN (Survivable Adaptive Radio Networks) program [25].

The main purpose of this plan was to provide packet switched networking for battlefield application. The main focus was achieving a better performance by making them smaller, cheaper and more power efficient, increasing the scalability of algorithms and resilience to the electronic attacks. The “Global Mobile” project known as “GloMo” project and “Near Term Digital Radio System” known also as NTDRS developed with unique characteristics like self-organization and self-healing for mostly military purposes for DARPA. [27]

There is another generation of ad-hoc networks (so called the third generation) which is the extension of commercial ad-hoc networks and was started with all the other technological revolutions like notebook computers with start of 20th century.

Two most important applications of ad-hoc networks, namely sensor networks and Bluetooth, are the products of this generation of ad-hoc networks. [27]

Generally ad-hoc networks and their challenges can be a good research topic. The topic of communication in an ad-hoc network between different hosts that are not connected directly is an interesting challenge and also an issue for search and rescue operations.

The focus of the current research is mainly an attempt to standardize different existing network controls for a single framework. This makes the use of these standards useful for future applications. Wireless devices are getting smaller as MEMS technology advances. So they also are getting cheaper and researches are to find a more cheaper way to keep these devices connected. [25]

If we want to consider ad-hoc networks in more detailed we have to study them in two major classification which are mobile ad-hoc network and mobile ad-hoc sensor network. [26]

2.4.1.1 Ad-hoc Sensor Networks

A mobile ad-hoc sensor also called hybrid ad-hoc network, is a network of sensors which are spread in a geographical area. Each member of the network or each sensor has the capability of detect and process different signals in the environment and transmit data through mobile communication. [26]

One specific advantage of mobile ad-hoc network is their ability to adaptability. The reason for this characteristic of ad-hoc sensor network is that the routing protocol determines if two mobile nodes are connected. So based on this, routing protocol routes packets between two nodes accordingly. All these are to support routed communications between two mobile nodes. This is the reason that a mobile ad-hoc sensor network is highly adaptable and it can be deployed in almost all kinds of geographical environments. [26]

The mobile ad-hoc sensor network as mentioned in 2.4.1 is one of the ad-hoc network categories which counts as a new invention. It's expected that this kind of ad-hoc network bring a huge range of applications that will transform our daily lives. In ad-hoc sensor networks, each host can use a large range of sensors for detecting different event in the environment of deployment. One of the major advantages of ad-hoc sensor network is the low cost for network setup and its administration. It's predicted that we will see a large number of this kind of ad-hoc network deployed for different applications in the near future. [26]

One main difference between ad-hoc sensor networks with typical sensor networks is setup process for the network. Typical sensor networks have a direct communication with the controller which is the type of centralized controller while a mobile ad-hoc sensor network follows a broader sequence of operational scenarios which makes it less complex in case of setup procedure. [26]

There are lots of benefits in different areas for mobile ad-hoc sensor networks. These kind of ad-hoc

networks are so beneficial in not only military applications but also civilian applications. As an example of their use in military applications is gathering information about all locations and movement which are necessary. [26]

As civilian applications there are mobile traffic sensor networks which can be used to monitor traffic on highways and motorways, and also a mobile surveillance sensor network that has been used for security issues in various places such as buildings like hotels. Mobile ad-hoc sensor networks can also be used to help for supporting required information for finding free slots in a parking area. These networks can also have different environmental applications like detecting pollution in oceans in advance and also an early notification about the start of firing in a huge jungle and prevent a huge environmental catastrophe. [26]

2.4.1.2 Ad-hoc Mobile Networks

As per definition and as a difference with other networking protocols, ad-hoc networks do not follow up the traditional network setups, instead this kind of networks will form a network with the help of an automatic system structure which is the center of an ad-hoc network. This is considered to be a revolutionary way of establishing a network which can be also an advantage of ad-hoc networks over all the other traditional network, specifically in environments with poor or costly to deploy, infrastructures. [25]

Only some years later, a force group was established which later on, ended up to finding new routing protocol standards specific to ad-hoc networks. The results of this working group, so called as IETF, ended up to settling new routing protocols for ad-hoc networks and therefore, promotion of this kind of networks. [25]

2.4.2 Ad-hoc Standards

Some of the IEEE 802.11 achievements for standardizing ad-hoc networks resulted in many industrial and scientific outcomes. Some of the most important achievements ended up to building prototypes which deployed ad-hoc networks on laptops, addressed ad-hoc networks in Bluetooth case and benefitted HYPERLAN in the same area. [25]

Some of the standardization achievements mentioned above, had extensive applications on itself, for example prototypes which built ad-hoc networks for mobile usage, was deployed in military use cases, medical care improvements and environmental applications. [25]

2.4.3 Autonomous Sensor Networks

ZigBee can be characterized as a type of autonomous sensor network. ZigBee is a set of rules under IEEE 802.15.4 which makes the data communication for WPAN (Wireless Personal Area Network) standardize. [31]

This networks point to the short distance usage of networking which may connect for example home appliances together. The main advantages of this network is the low power consumption, low cost, high density of nodes in a ZigBee model and the simple protocol that makes the implementation of the ZigBee models easier. ZigBee also suggests the good and acceptable level of security and is so reliable in case of data transmission. This technology could be an attractive type for short range applications. [30]

ZigBee network consists of different layers which are physical layer (PHY), medium access control sub-layer (MAC), network layer (NWK), application support sub-layer (APS) and finally application layer (APL). Application layer consists of the ZDO module which is ZigBee device object. A security service provider (SSP), which is in contact with NWK layer and MAC layer, provides an appropriate level of security for the network. By using the self-organizing concept or other models for ZigBee, it can have very interesting applications and handles different type of data traffic pattern. [30]

This network can take benefit of different topologies or a mixture of two or more topologies like mesh, star and cluster tree topologies. [30]

There is this idea that ZigBee serves the similar application as Bluetooth does but there are some major differences that makes them special for some special range of applications. As a comparison in terms of air interface, it can be seen that ZigBee uses DSSS while Bluetooth uses FHSS. Also in terms of power

consumption, ZigBee performs a faster series of activities which let it to go to sleep and stop losing energy as soon as it does its work. So it can provide lower power consumption and longer battery life time in the network it has been deployed in. These differences make them to have different range of applications. For example ZigBee would be a proper choice for sensor network applications or in the applications where battery life time is important. [30]

3 ARTIFICIAL INTELLIGENCE FOR SMART DUST NETWORKS

3.1 Introduction

It should be known if intelligent algorithms in soft computing area like Neural Network, Genetic Algorithm and so on, are useful for improving the smart dust network performance. As we know most of the smart dust networks which supposed to be deployed for applications in large scale areas like forests, should have the ability of self-organizing; so they are also in the category of self-organizing networks (SON) as well as Smart Dust networks. We also know that SON is an improved version of Artificial Neural Networks (ANN) which has different features like unsupervised learning. So it already has the properties of an ANN in itself. [4]

There have been a lot of attempts to use this kind of soft computing algorithms to improve the characteristics of wireless sensor networks and smart dust networks as a part of them. These researches were mostly concentrated on using a soft computing algorithm in this kind of network, so that it results in to decrease the power consumption of the network and so increase the life time of the motes.

For example Berkeley has improved a multi-hop communication approach which fairly distributes the energy consumption over the motes. Chen et al. developed a coordinate algorithm to increase the energy efficiency which has a distinctive feature call SPAN. So based on this algorithm, motes can decide whether they should stay awake and communicate actively in network (which will consumes power) or they should sleep (and stop power consumption); and these states for an active mote changes continuously. [6]

In this chapter we will try to select some famous soft computing algorithms and point out if they can help smart dust network in terms of performance improvement. It has tried to point out how an algorithm can be helpful in this case and if a soft computing algorithm won't affect this process, the related reasons has mentioned for it.

First we will try to give and overview our selected soft computing algorithms in chapter 3.2. Then we will give an introduction on the selected algorithms and will discuss if the mentioned algorithm is useful for being deployed in smart dust network and discuss the reasoning why they are beneficial to be used or they are not. Section 3.2.1 describes Fuzzy Logic. A whole detailed overview on self-organizing Networks can be found 3.3.

3.2 Artificial Intelligence

Almost from the time computers and other intelligent digital devices started to grow in the world, scientists wanted to make them more and more advanced. There have been a lot of research on how a system can behave more closely to human brain. How it can decide better and how it can be possible to increase the machine intelligence (MIQ) based on human brain.

3.2.1 Fuzzy logic

As for the Fuzzy Logic (FL), the most useful application which has reported using this technology is related to control applications. [3] There are some application for controlling the traffic in a network which Fuzzy Logic was quite successful in that. [4] So it is believed that it can be useful in smart dust networks as well, but the estimation is that it may not improve the energy consumption at this stage distinctively. [5]

3.3 self-organizing Networks (SON)

A self-organizing network is basically referred to a cellular network which different tasks such as configuring, operating, optimizing and healing are automated. Their target is to reduce the operational costs and providing good user experience even in critical conditions such as traffic congestion. SON has also energy saving features that can contribute in a greener network environment.

3.3.1 Introduction

With ever increasing trend in cellular technologies, different aspects such as optimization, planning, management and configuration of the system have to be automated to improve the performance. Therefore, the concept of Self-organizing Networks became more popular. These networks are able to monitor, organize and optimize themselves; and in case of occurrence of a failure, heal themselves. Operators can have benefits on the improvement of CAPEX (Capital Expenditure) and OPEX (Operational Expenditure).

This rapid growth in cellular and mobile communications happened with introduction of 4G/LTE technology, where there was a considerable increase in data usages. Radio network planning and maintenance became more complex; networks became more dense and complicated.

Later on, 3GPP (Third Generation Partnership Project) introduced the concept of self-organizing networks in the eighth release of their standard in December 2008. Similarly, NGMN (Next Generation Mobile Networks) has also introduced SON concept. The aim of this was that by having less human intervention in design, manufacture and operation of the network, the operating costs will be decreased; by having less user errors, the revenue can be protected; and by optimizing the usage of resources, capital expenditure can be decreased.

The first efforts in developing SON networks were dedicated to radio access elements, because they are holding a major part of the costs for the installation, deployment, and maintenance of the network.

There are two main reason for automating a cellular networks, which are:

- *The user experience:*
These networks has two important capabilities that are quick optimization of the network and mitigating any occurring outages, causes improvements in user experience. These capabilities are important for the network because of the two critical factors of time-to-operation and time-to-repair. Load balancing between nodes in e.g. a congested traffic is another strong points of SON. It can also prevent or minimize user overloads by distributing bandwidth among them.
- *Operating efficiency and cost:*
In case of operation efficiency, the goal is to simplify, run, maintain and optimize the network in an efficient and autonomous way. This is very useful when considering the complexity of such technology. It also increases the life cycle of the network.

A huge boost in usage of packet networks changed the pattern of data traffic. It all shifted from voice traffic to data traffic. Considerable popularity and wide spread use of handset devices like smartphones and tablets, having access to thousands of applications and internet services anytime, and anywhere made users to have higher expectations. This causes a fluctuation in bandwidth and QoS requirements of the system. In order for the network to be able to handle this requirements, it should adapt to dedicate the same amount of bandwidth to all the users, optimize load distribution among the cells and guarantee robust mobility and handovers.

3.3.2 SON Architectural types

There are three main architectural types of self-organizing networks, listed below:

3.3.2.1 Distributed SON

In this type (D-SON), the functions are distributed over the network elements. Hence, there is some level of

localization functionality which is supplied by the vendor who manufactures the radio cell.

3.3.2.2 Centralized SON

In centralized SON (C-SON), in order to have a more broad view of edge elements and coordination, functions are more closer and concentrated to higher order network nodes. C-SONs are usually supplied by third parties such as Cisco or Celcote.

3.3.2.3 Hybrid SON

H-SON is a combination of centralized and distributed SON, utilizing elements of each in a hybrid solution.

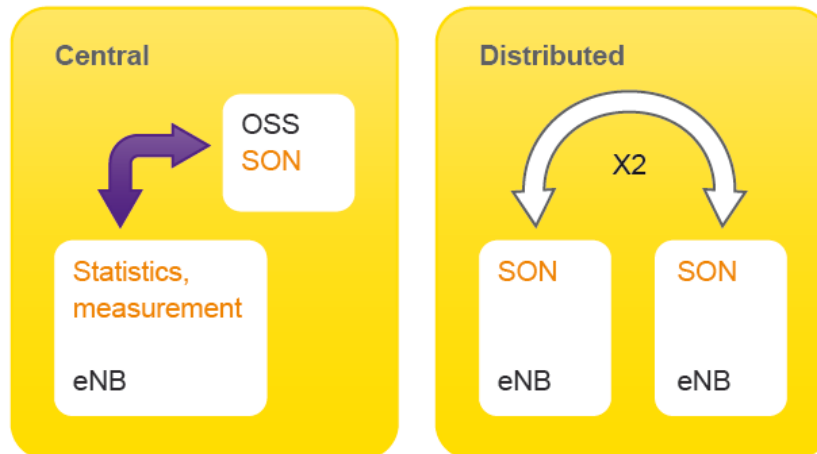


Figure 2: The centralized and distributed architecture type of SON network

3.3.3 SON Functions

There are three main functions associated to self-organizing networks that are described briefly below:

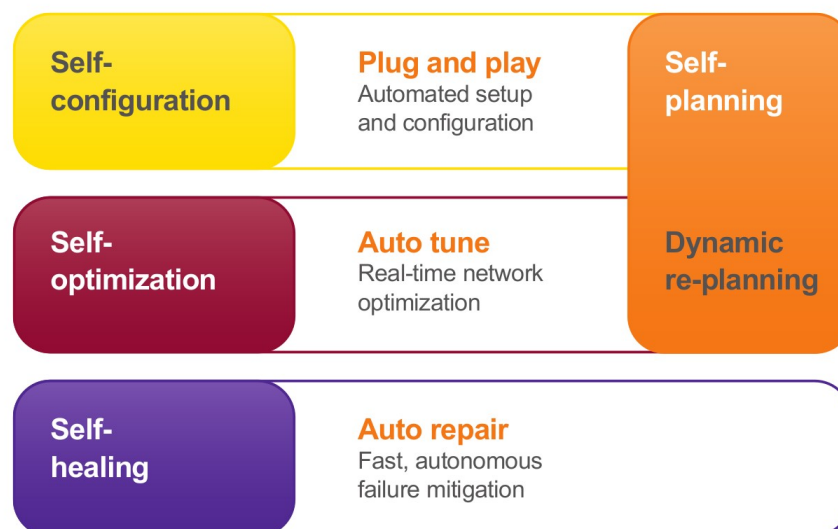


Figure 3: A review of SON functions

3.3.3.1 Self-configuration

The goal is to have all the base stations as “plug and play” items that can configure themselves with a few necessary manual settings. This means no need for professional installations skills and less cost. Therefore this is an important element of the SON network.

3.3.3.2 Self-optimization

Self-optimization is essential after setting up the system; to optimize the operational characteristics and satisfy the needs of the network.

- *S e l f - p l a n n i n g*
This function combines configuration and optimization capabilities to dynamically re-compute parts of the network, and to improve parameters that are effective in the service quality.

3.3.3.3 Self-healing

Faults are inevitable in any system, and can create problems for the user. So, the whole network can temporarily hide the effects of the fault by changing its characteristics. To cover the area that the fault occurred, neighboring cell can expand their boundaries by increasing their power. Moreover, auto-restart and automatic alarm feature will allow the user to get more faster response from the network. The self-healing function of SON helps to compensate for outages, and to repair any failures much faster and easier. It also provides temporarily compensation for equipment outages, until a more permanent solution is found.

Newly added base stations should be self-configured in line with a "plug-and-play" paradigm, while all operational base stations will regularly self-optimize parameters and algorithmic behavior in response to observed network performance and radio conditions.

3.3.3.4 SON Self-Configuration

This feature allows a new cell to be added to the network with a “plug and play” method. It reduces the costs and makes sure that all the cells are integrated correctly to the rest of the network. It is of great importance to make sure that this process is automated as much as possible. Since cellular networks are becoming more complicated, cells are becoming smaller, revenues per bit decrease; therefore, it will be difficult to handle the increased level of overall data traffic manually.

There are a couple of features involved in the self-configuration of a new base station, which are listed in the following:

- *Automatic configuration of initial radio transmission parameters*

This parameter is an essential factor while the self-configuring process. In some cases, it is better that the base station gathers the data itself, and by considering the fact that the data is not exactly as expected, some adjustments might be necessary which are very time consuming.

In such cases, the Dynamic Radio Configuration (DRC) technique is used to adapt the base station to the current topology of radio network. In this way, one can make sure that instead of the estimated values, the real measured values are used.

- *Automatic neighbor relation, ANR, management*

It is an important and heavy activity for mobile network operators to have the correct relationship between the adjacent cells, as this helps in performing easy handover. When having wrong relationship, a call can be dropped.

In case of manual updating of the neighboring connections, the network has a much more complicated task to see if the cell can handover to its neighbor cell with the same radio access technology, or has to change it for example from LTE to HSPA or else. The base station provides a neighbor list of all the relationships. Updated and optimized lists can boost network performance, increase the number of correct handovers and decrease the network load.

- *Automatic connectivity management*

The automatic connectivity enables the base station to automatically connect to its domain management system. For setting up this connectivity, there are the following stages:

- Basic connectivity set-up
- Initial secure connection set-up
- *Site identification*
- *Download of final configuration and transport parameters*
- *Secure connection set-up*

- *self-test*

A self-test is done to make sure that the equipment are working correctly before the final active service.

- *Automatic inventory*

Automatic inventory involves different activities that enables the base station to understand its functions; activities such as identifying hardware boards, software level, antennas etc. Since different base stations have different capabilities, it is essential for the SON self-configuration software to run an inventory check first.

3.3.3.5 SON Self-Optimization

To make sure that a cell is operating at its best with high efficiency, optimization is necessary. Self-optimizing techniques are used for performance analysis and matching network operation to the user needs.

Self-optimizing networking techniques can be utilized to address some changes that can happen even after the network is configured.

3.3.3.5.1 Self-optimizing motivation

Self-optimizing techniques are very important features of the self-organizing networks. It is likely that changes occur in the environment of the base station even after installing and configuring the network. Hence it is necessary to optimize the operation more regularly. Below, some reasons for environmental changes are listed:

- *Change the propagation characteristics*

Any changes in the environment can lead to changes in propagation. Building new buildings, taking down one, the traffic, and even falling leaves can have considerable effect.

- *Change in traffic patterns*

During different times, usage patterns may change. There might be higher concentration during holidays, school vacations, or when a new housing area is built, bringing more users into that section and many more reasons. Other causes can decrease user traffic concentration. Thus, we need further optimization to find the optimum operation for the network.

- *Change in deployments*

Deployment of other base stations, eNB will have an effect in changing the environment. Other base stations can optimize or vary their characteristics, and hence new base stations should optimize and adapt themselves accordingly.

3.3.3.5.2 Self-optimizing network functionality

In the following the use of self-optimization networks in some areas are discussed briefly.

- *Mobility robustness optimization*

One of the features included in self-optimizing networks that tries to establish robust mobility and handovers in the mobile network. There are some reasons for using this technique, listed below.

- *Minimized dropped calls*

Dropped calls are one of the main reasons that customers become unsatisfied. So, reducing them by improving the network is a very important factor.

- *Minimized unnecessary handovers*

Unnecessary handovers use the resources inefficiently and causes the calls to drop. This often happened in the boundaries between two adjacent cells that several handovers take place by a small change in the position.

- *Minimize idle mode problems*

The handover should setup the connection, right after an idle mode.

- *Minimize radio link failures*

Radio link failures are something that can happens several times. The aim of the self-optimization is to avoid these failures by providing good coverage. In case a failure happened, the link should be able to reconnect again quickly.

- *Mobility load balancing and traffic steering*

Usually, some of the cells are like data hotspots and have a heavy load on them comparing to other cells. Leveling out these data hotspots is what Self-organizing elements do and it is called load balancing. Load balancing elements distribute the load efficiently among all nodes, utilize the right resources while keeping good capacity and investment levels.

Load balancing management is becoming a more essential task as the network traffic rises rapidly. However, for undertaking this task, there is a need for complicated routines in the SON network.

In the following, some of the tasks in the load balancing and traffic steering software are described.

- Data traffic is shifted from a cell with heavy load to a less loaded cell to level out the load and reduce data traffic.
- Data traffic is transferred from macro cells to smaller and low power cells like HeNB and WiFi.
- To gain maximum performance, those mobile handsets that are moving are not assigned to smaller cells, because they will move away from the cell and cause a lot of handovers. Hence, the self-optimization software should be able to detect any kind of movements in the network.

- *Energy saving*

Energy saving and the use of green energy is a crucial matter in almost every technology. The idea is inspired from the need for reducing the carbon dioxide emissions and of course by reducing power consumption, we'll gain cost savings as well.

Energy savings can be made from both mobile handhelds and also the network. In case of SON self-optimization, the main saving activities are done within the network, particularly in base stations, eNBs.

There are different methods of performing energy saving in self-optimizing networks. Basically, there are no special features incorporated in the network for this purpose, but in some circumstances, considerable power saving can be achieved. For example, during the night, when the traffic is much lower than during the day is a good chance of saving significant amount of energy.

A couple of options suggested for utilization of energy saving methods are:

- *Reducing active carrier for off-peak times*

In order to meet the data capacity requirements, base stations send out several carriers that can be reduced during off peak times, resulting in power reduction.

- *Sleep mode*

In some areas like business related sections, comparing to daytime, there is a very much lower usage levels (close to zero) during night time of weekends. Thus, it is possible to have some of the base stations in sleep mode and increase the coverage on the remaining stations. However, it is essential not to leave holes in the network causing trouble and dissatisfaction for any existing user. The base stations should also be able to wake up from the sleep mode quickly.

- *Local generation*

With renewable energy resources that have a very low carbon footprint network (such as solar or wind energy), there are more options for local generation. However, mains or grids are being used as power source by most of the base stations.

- *Coverage and capacity optimization*

The coverage and capacity optimization, CCO, can be done in a couple of ways, listed below. It deals with adapting some parameters like transmitter power levels, antenna tilts, and so on to increase the network coverage and optimize the capacity. Although it can be very time consuming and expensive (if done manually), it can have some advantages as well.

- Adjustment of antenna parameters
- Adjustment of power level parameters

- *RACH optimization*

The Random Access Channel, RACH, is an essential part of the access scheme that consumes valuable resources. Therefore, a balance should be maintained in the trade-off between dedicating resources and compromising performance.

There is also the need for an optimized network at all times to fit the condition changes. This can be done in two different mechanisms: “Handset reporting” and “Inter base station data exchange”.

Several techniques are being employed for SON self-optimization networks to achieve the necessary functionality that are not simple and needs a lot of investment from the operator.

3.3.3.6 SON Self-Healing

This feature of the self-organization networks is becoming more and more important, as it can detect faults and hide their effects to users, while the repair process is undergoing. It is more important that the self-healing network make sure that the whole network operates properly, even in presence of a fault.

As cellular networks are expanding size and are becoming more complicated, it is obvious that failures will occur. Most of the faults can be noticed by the users; while some can't.

3.3.3.6.1 Self-healing network basics

Faults can emerge in any part of the network. In some cases, the fault can be bypassed without a serious issue, but in some other cases, we may need backup hardware.

The part of the radio access network which is more sensitive to faults is the base station. Any fault or service loss within the base station, results in considerable performance degradation, causing significant revenue loss for the operator.

There are some areas that self-healing concept is used. A list of those main areas are given below:

- *Self-recovery of software*

Ability to return to a previous version of the software

- *Self-healing of board faults*

Use of redundant circuits where a spare can be switched in

- *Cell outage detection*

Possibility of remote detection when there is an issue with a particular cell

- *Cell outage recovery*

Routines to assist with cell recovery, along with an automatic recovery solution, and a report of the outcome of the action

- *Cell outage compensation*

Maintaining the best service to users while the repairs are being effected

- *Return from cell outage compensation*

Possibility of easy return to the pre-fault status, removing any compensation actions that may have been initiated

To perform the self-healing action, there are some techniques such as built-in testing, monitoring methods, data collection and analysis that is incorporated this functionality into the network to enable detection and managing the faults.

3.3.3.6.2 Cell Degradation

The first step to self-healing is to detect the fault. Without knowing that there is a fault in the network, it is not possible to address it. So, the performance of the base station, as well as the measurements of some key performance indicators, KPIs, such as output power are being monitored. In turn, the key performance indicators are also under monitoring to make sure that they are not having any problem.

At the moment a fault is detected, an alarm will be flagged to the operation, administration and management center and a manual or automated action is issued.

In any fault detection scheme, there should be limits or threshold levels for acceptable measurements. If the limits are set too broad, detection is not possible; narrow limits also give false alarms. Hence, it is necessary to optimize the limit as well.

3.3.3.6.3 Cell outage compensation

Cell outage compensation determines an important feature of the self-healing network in occurrence of a cell outage. There is an essential requirement for this compensation that has to be carried out automatically. And that is that the overall network should act quickly in detecting the fault and assessing its impact in order to apply the compensation.

Sometimes the cell outage compensation comes from the adjacent cells. This is done by expanding the are of the adjacent cell so that it covers the faulty cell. Techniques such as antenna tilting and increasing the base station power are used to achieve coverage and capacity optimization.

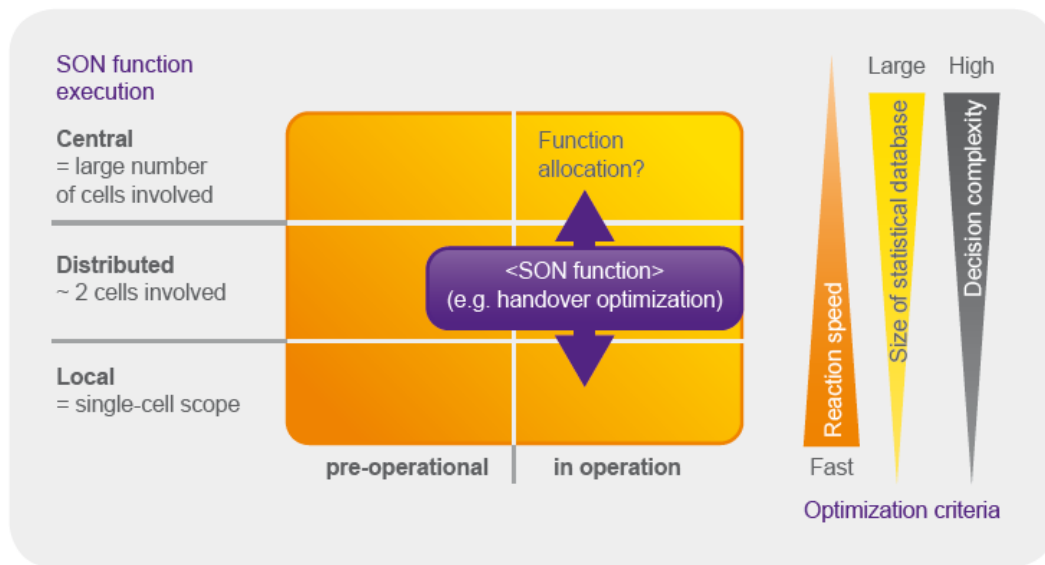


Figure 4: Different key factors of SON functions

In conclusion, self-healing techniques can be used to cancel the effect of a cell failure. However, still there will be some effect on the performance. So, the network should respond quickly and return to its conditions before the fault.

4 SMART DUST NETWORKS

4.1 Introduction

Smart dust is a network of several very small devices called 'mote'. They are mainly MEMS devices (Microelectromechanical systems) consisting of sensors and other electronic devices. Sensors can be from a wide ranges of types that can detect temperature, moisture, light, magnetism, chemicals or vibration from their surrounding environment. Motes can be distributed over an area for performing a specific task and are usually communicating through a wireless network. [1]

The initial concept of smart dust was originated in 1992 from a workshop at RAND and also some studies done by DARPA in mid-90. They were interested in this technology because of the foreseen applications in military. UCLA and University of Michigan were the two main partners that influenced the project. [1]

In 1998 DARPA granted a research fund for a project presented by a bunch of scientists from UC Berkeley lead by Kristofer Pister from UC Berkeley, to build a smart wireless sensor nodes smaller than a cubic millimeter. They were actually successful in building a working node in the size of a grain of rice. Later on Pister expanded the concept even more. [1]

4.2 Opportunities and challenges

In terms of possible opportunities and applications, the examples where this technology can be utilized is simply unlimited. For example one can sprinkle millions of tiny wireless sensors over the oceans to get a better data of the health of the oceans. Sensors float on water and can communicate to each other and send the data through radio waves. They even don't need any battery since they can collect their power from the motion of waves that will constantly charge them. Researchers can only check them in case of occasional maintenance. [33]

One can incorporate them in asphalts for detecting the vibrations of passing cars to be able to monitor the traffic in a much efficient way. Utilizing this technology in farms and croplands gives the farmers a better understanding on how to save water and use less fertilizer and in turn save money and increase their yield. [33]

By spreading wireless seismometers all around the world, it might be possible to detect earthquakes so that people could get a 30 to 60 seconds warning before anything happens. This time might not seem to be long enough, but it is sufficient for people to get out of buildings, hide or duck under a table and for buildings to disconnect the power or sources that may blow up later. This decreases the chance of having high casualties. [33]

It is known that with every new technology comes challenges and opportunities. In the case of smart dust, considering the small size of motes, it is not possible to mount a large antenna and hence the range of communications between motes are measured to be very small e.g. a few millimeters. There is also a considerable chance that the motes be destroyed by microwave exposure or other natural phenomenon. [1] As a solution one may refer to the 'dust' feature of the motes and the fact that they are very small. They can be spread over the desired area in high numbers so that if some of the motes are destroyed or by any reason go off grid, there are other motes to do the job. [1]

One may also argue the risks or dangers these devices may have to the nature and wild life inhabitants. [1] After all this, there is a huge argument that this technology is a huge potential threat for privacy of

individuals. We are talking about many tiny wireless sensors that can easily be undetectable. Therefore, there are some issues and arguments regarding the smart dust among researchers and users.

4.3 Smart Dust Mote Architecture

A smart dust mote can be consisted of different components, which a sample is shown below, in Figure 5. These components can be different, based on mote's application in the smart dust network.

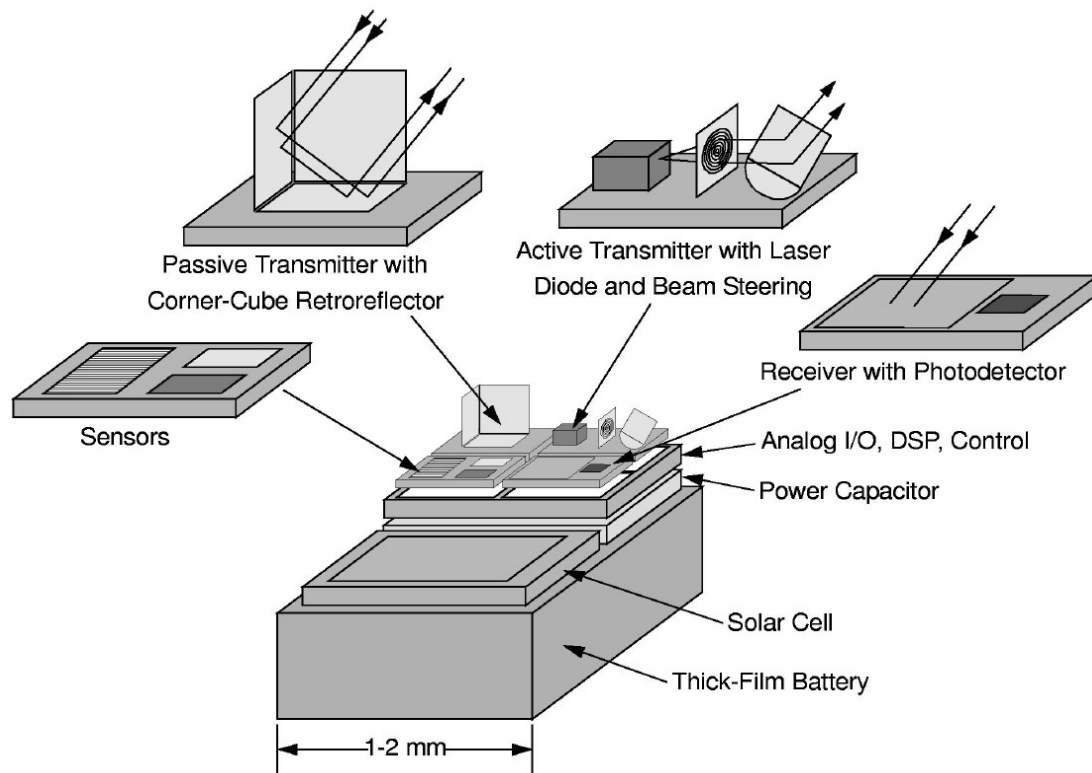


Figure 5: A smart dust mote, containing different parts such as sensors and other network equipment. [35]

Figure 5 illustrates the components of a smart dust mote fabricated using MEMS technology. The laser diode and beam steering mirror are used for active optical transmission, and the corner cube retro-reflector is used for passive optical transmission. If the incident light is within a range of angles (centered diagonally), then the light is reflected back to the source.

4.4 Power management strategies in Smart Dust

There are different approaches to reduce the total power consumption in wireless sensor networks. Based on the circuit techniques that can be used to decrease the amount of energy that the network is consuming, one can categorize them as follow:

- *Sub-threshold operation*: Some of the systems in the University of Michigan use a power supply lower than threshold voltage to reduce the active power consumption by sacrificing performance.
- *Asynchronous circuits*: SNAP, a processor from Cornell University eliminates the clock power by using asynchronous circuits.
- *Power supply gating*: In order to deal with the problem of leakage current, transistors can be used

to turn off the power supply to those blocks that are not being used. This is done on systems from Harvard University and the University of California, but each use different method to do this.

- *General purpose computation:* which is based on using processors with load-store or accumulator in the center of the system and as the core processor.
- *Event-driven:* In this category, network responds only if an event is detected in the environment which need the attention of that network.
- *Applications acceleration:* Using hardware acceleration in the system will also decrease the power consumption.

4.4.1 Harvard Event-driven Architecture

As it is mentioned above Harvard system has utilized hardware acceleration to effectively improve the performance of common tasks in applications for sensor networks. It is also using power supply gating control to solve the leakage current problem. This system uses three trend to decrease the power consumption: [36]

- *Lightweight event handling in hardware:* A specific event processor has the initial responsibility to handle the incoming interrupts. This eliminates the software overhead.
- *Hardware acceleration for common WSN tasks:* Message routing and data filtering and other common tasks are done using a modular hardware accelerator.
- *Application-controlled fine-grained power supply gating:* This will deal with the issue of leakage current.

4.4.2 Low power design

The ultimate goal of designing a smart dust mote is to integrate several electronic components (such as MEMS sensors, optical transmission and receiver, thin-film power devices, signal processing and control circuits) into a very small chip while have a very low power consumption in order to increase mote's life time. The mote also has to be able to provide its own power from different ambient sources, such as solar energy. And of course all the electronics should be designed for low power conditions. [35]

Storing power can be a tough challenge since there is not much space on the chip (with volume of a cubic millimeter) for battery or storage devices. Even with the best battery technology available, the amount of energy that can be stored on a mote with mentioned volume, will be in the order of 1 Joule. Assuming that the mote consumes power continuously, the maximum power consumption of the mote will be less than 10 Watts, which is not a considerable amount at all. In fact this is the power used for shutting down a low power IC in your laptop computer. The power range that we are aiming for in the smart dust project is of the scale of micro watts. This is the reason power management policies are being utilized to maximize the efficiency of power usage. For example a part of the device turns on whenever necessary and it is off when there is no task for that section. [35]

For collecting energy, dust motes benefit from energy harvesters or energy scavenging circuits that can gain power from several sources of energy in nature. Energies can be gained from solar energy, vibrations or movements, winds, waves and so on. For instance solar cells used to store solar energy can harvest about 1 joule energy each day, when the sun is shinning, and about 1 mJ per day when the room lights are on. With every joule that the dust mote stores, it can perform a huge number of computations. The challenge here is how to allocate this power in an efficient and effective way. Energy cost is another subject that is important. [35]

In respect to low power design, sense and computation techniques at low power are well known. The challenge here is to design a low power communication architecture. Currently the candidates for the communication schemes are RF communication or optical transmission techniques. Each of these techniques has their own pros and cons. The problem with RF is that due to small dimensions of the dust mote, we are limited to mount small antennas, which means working in significant short wavelengths

transmission. This also impose low power operation in high frequency ranges, which is not fully possible yet. Using high frequency means that we cannot be in the low power operation. Moreover, complexity of radio transceivers makes it difficult to lower the consumption down to microwatt levels. [35]

4.5 Sensors used on dust motes

Based on what is intended to be detected, different types of sensors can be used for sensing that factor. There are many types of sensors available to detect parameters such as temperature, moisture, vibration, acoustics, magnetic field and wind shear. Sensors will be integrated onto the mote using the MEMS technology. With all the advances that have been made in area of MEMS technology, we are witnessing exponential decrease in cost, size, power consumption and performance of these devices.

Sensors used in dust motes can be divided into sensors which detects the movements in an environment and sensors which control the weather and environmental figures. Table below, shows some of theses categories.

Table 1: Category of Sensors types [40]

	Application
Magnetometer	To measure the magnetization and its direction and also to measure
Accelerometer	To measure acceleration power
Light sensor	To perform light detection
Temperature sensor	To measure temperature
Pressure sensor	To measure the pressure
Humidity sensor	To measure humidity in an environment

4.6 Communication methods in Smart Dust

One of the most important aspects of the Smart Dust network is the communication between dust nodes. All the motes in the network have to communicate with each other through the base station. While considering all the design constraints due to size and power limitation, data must be collected form the motes simultaneously, sent to base station for further action.[40]

In down-link (i.e. data propagation from base station to the dust motes), base station broadcasts to all the motes in the network at a rate of several kbps. And in uplink (i.e. data propagation from motes to the base station), the data transfer rate is of 1 kbps. Hence, if a total number of 1000 dust motes are employed in the network, the data throughput will be 1 Mbps. The data transfer both in uplink and down-link should support distances of a couple of hundred meters. [40]

There are other specifications regarding the mote. Dust mote size should be less than 1 mm^3 and must have a power consumption of at most $1 \text{ }\mu\text{W}$. We also need a secure and reliable transmission method for communication in the network. There are several options possible, but in this thesis work, we will discuss two methods: Radio frequency (RF) and optical communications. Each of them have their own strong and weak points that is discussed and compared in the following section. [40]

Basically, the task of communication system is to send and collect commands to and from motes. We can categorize them into following groups:

- Radio Frequency Transmission
- Optical transmission technique
 - Free-space optical communication

- Passive Laser based Communication
- Active Laser based Communication
- Fiber Optic Communication

4.6.1 Radio Frequency transmission

RF technology has advanced a lot in these years and is being used widely in different applications. In this method of transmission, radio frequency signals of range from tens of Kilo Hertz to hundreds of Giga Hertz are used. RF can be used for both uplink and down-link. [40]

RF communication is a very good potential candidate for Smart Dust networks, but at the same time, there are a couple of problems associated with it. Some of them are mentioned below: [40]

- Due to the complex circuitry in RF transceivers, it is almost impossible to achieve the low power specification requirements needed for Smart Dust system.
- One of the issues addressed in antenna design is the size limitation. Size of a fabricated antenna is limited by margins that the range of cubic millimeter puts on it. An antenna size cannot exceed a quarter of wavelength of the carrier. That means that this size shall be in defined in the area of very short of wavelength which on its own result in having an operation which not necessarily runs with efficient power consumption.
- Due to the high number of dust motes, it is necessary to use multiplexing techniques for achieving RF communication. Multiplexing techniques such as code-division multiplexing or time/frequency multiplexing. This means that several kinds of RF circuitry like filters, modulators and demodulators, are needed and they all should be designed for low power consumption.
- Using multiple access techniques such as TDMA or CDMA or similarSDM, has their own complexity which is not compatible with Smart Dust system.

4.6.2 Optical communication techniques

Optical communication utilizes semiconductor lasers and diode receivers for transferring optical signals. In comparison to RF communication, this method is more compatible with the low power design requirements due to the small size of optical transceivers. one can tune the optical power to a focused and localized beam. In optical communication, we can easily create a 1 GHz signal from a millimeter aperture, but to produce the same signal in RF communication needs an antenna of 100 meters (due to the wavelength difference between two transmissions). Optical transmitters can also have gains even more than a million. [41]

With respect to power, once again, optical communication has the advantage. One reason is that optical transceivers have much simpler circuitry. [41]

4.6.2.1 Optical communication – Main category: Free-Space

In this case, we need very narrow beams and line-of-sight which are two drawbacks of this communication method. However, by exploiting smart and effective algorithms we can gain accurate pointing; and using MEMS technology, we can achieve very narrow beams. So, in applications where there is a line-of-sight free space optical communication links can be used with much less energy usage comparing to RF. Now two ways of optical communication will be discussed: The active steered laser system and passive reflective system.

4.6.2.1.1 Optical communication – Sub-category of Free-Space: Active and Passive

In this approach an active laser diode is used to transmit a focused laser beam to the base station. This system is consisted of a beam-steering micro mirror, a semiconductor laser and a collective optical lens. All these components can be fabricated onto a chip within the size range of Smart Dust motes. [42]

A drawback with this system is that it uses a lot of energy. Therefore, this system should be used in very short burst mode communication to avoid using much power. It is also very important to have a directional

beam aiming exactly towards the receiver. This results in high design complexity of the dust motes. [40]

In contrast to active optical communication, the passive optical communication uses a micro fabricated corner cube retro-reflector (CCR) and it does not need an on board laser transmitter. A CCR is consisted of three mutually perpendicular mirrors made from polysilicon with a coating of gold. [43] Any incident light in a specific range of angles hitting the center of the cube's diagonal will be reflected toward the source. If any of the three mirrors in CCR is misaligned, the light will not be reflected back. The incident light is modulated in the range of KHz when the electrostatic actuator in CCR deflects one of the mirrors. Since the dust mote is only deflecting or reflecting the light, there is no need emit any light and thus, it consumes much less power.

In order for CCR-based passive optical links to send and receive data, they need to have a clear line of sight. CCR can transmit data only if it is pointing directly towards base station within a few tens of degrees. One can use several CCRs pointed to different directions to increase the chance for communication. But this will need more space on the chip, meaning larger mote size. That is the reason size limitation of a dust mote is very challenging. Fitting all these devices and a battery on a single chip is very challenging in respect to size limitations, energy consumption and weight.

A bidirectional free-space optical communication link is one way. For the down-link the optical transceiver in base station emits an on-off-keyed signal to where there are a bunch of dust motes using a laser transmitter. This beam contains the down-link data. And on each dust mote there is a photo diode, a bandpass optical filter and a preamplifier. All this process is done in low speed base-band limit. [40]

At the same time, there are some limitations associated with passive optical communication. This method has a single-hop network topology and can only be used for communication between dust motes and base station, and it will not support peer-to-peer communication between dust motes. Also since the dust motes are dependent on the light source from the base station, in presence of a clear view, the mote is dedicated to that communication path and cannot do anything else, i.e. it will be isolated from the whole network. Another drawback is the waste of the illuminated light, due to the fact that the CCR component only reflects a small fraction of the incident light beam. The communication may also face variable delays if the source beam is not already pointed or aligned with the node subjected to transfer data. [40]

4.6.2.2 Fiber-optic communication

In this approach, a semiconductor laser, a fiber cable and a diode receiver is utilized to generate, transfer and detect the optical signal respectively. Most of the features are the same as passive optical communication, since similar techniques are also used in fiber-optic communication. The power consumption is low due to the small size of the optical transceiver. Same as before, there is no need for a light source on the dust mote itself, CCR modules on every mote will modulate and transfer the data to base station. [40]

There are some advantages and disadvantages in the fiber-optic communication in comparison with passive optical communication. In fiber-optic communication there is no need for having the line-of-sight, because it uses fiber-optics to transfer and receive the optical signals. This method is also safer for human eyes because no laser is involved. Longer range of communication and guaranteed communication between the dust motes and base station are two other advantages.[40]

The fiber-optic cables are the source of the limitations that will limit the movement and mobility of dust motes. Moreover, each dust mote should have a connection to base station (lots of cables), which makes the design of base station a complicated task. [40]

4.7 Routing Protocols

Routing protocols in wireless sensor networks can be categorized based on two main factors:

- Based on network structure
- Based on protocol operation

Based on network structure, we will have three different routing protocols: [20]

- Flat network routing
- Hierarchical network routing
- Location based routing

Based on protocol operation will give more different routing groups than network structure. There will be five different routing available based on protocol operation which are consist of: [20]

- Query based routing
- Negotiation based routing
- Multi-path based routing
- QOS based routing
- Coherent and non-coherent routing

Routing protocols may also be classified in another category which mainly focus on the way of communication and transmission between every sensor node and base station. Based on this view there are three possible models: Single hop model, multi-hop model and cluster-based model. Description for every of the mentioned models from the latter category can be seen below.

4.7.1 Single Hop Model

Single hop communication as it can be seen in Figure 6 refers to the type of communication that every mote communicate directly with the base station and send the sensed information without using a medium. This model has both advantages and disadvantages; for example, this way of communication is the simplest model to reach a base station and also the fastest way to report to the base station or the sink node. In the other hand, this kind of transmission is highly unrealistic in the real world. [19]

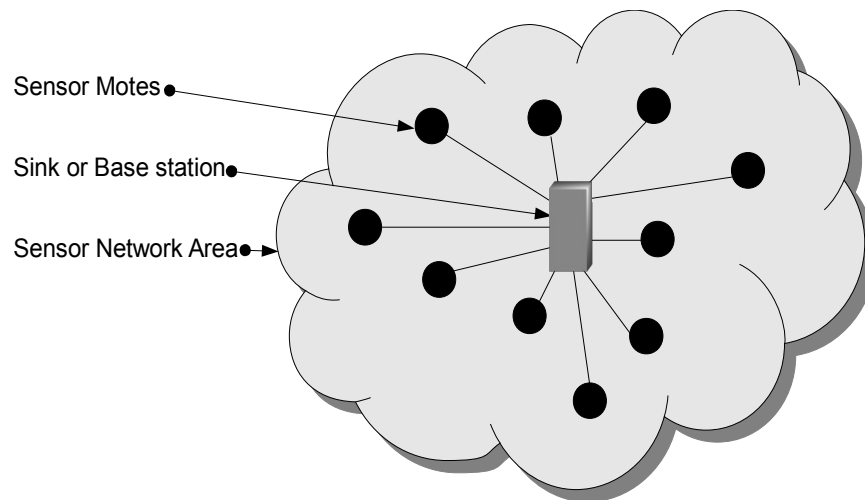


Figure 6: A simple view of single hop communication routing model

4.7.2 Multi-hop Model

The multi-hop model supports the collaborative effort of several nodes within the sensor cloud. each sensor node has a radio range, which is referred to as the distance which the signal strength remains above the minimum usable level for that particular node to transmit and receive data. [19]

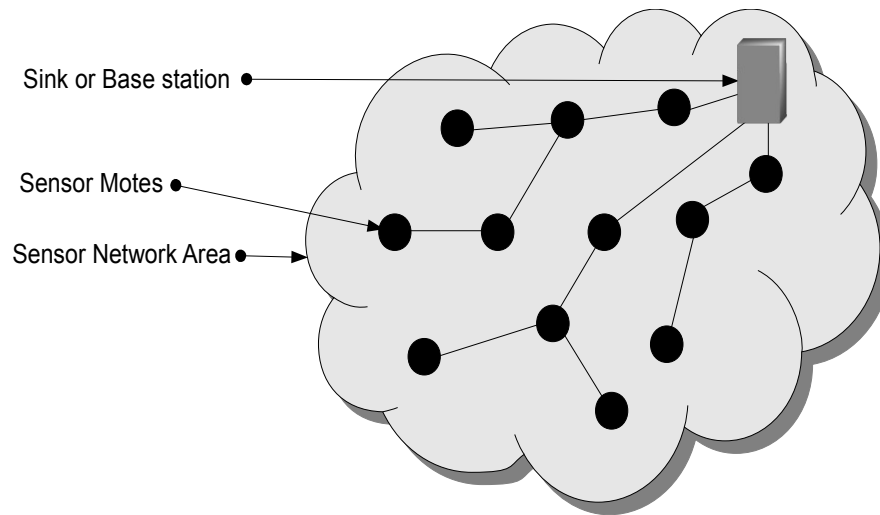


Figure 7: A simple view of multi-hop transmission in wireless sensor network

4.7.3 Cluster-based Hierarchical Model

In the cluster based model, the network is divided into clusters, each comprising of a number of nodes. Cluster head, which is the master node within each respective cluster is responsible for routing the information to other cluster head. [19]

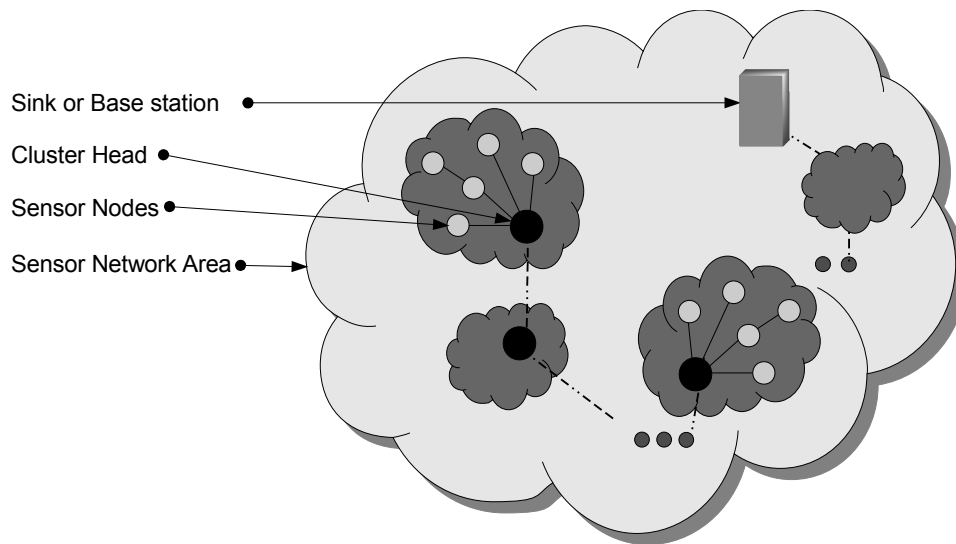


Figure 8: A view of cluster based routing model

There are two main types of clustering possible for sensor networks which are uniform clustering and non-uniform clustering. Below a short description of each is given.

4.7.3.1 Uniform Clustering

In this type of clustering, as we can see in 4.7.3, sensors will be grouped in different groups. Each group consists of the same number of sensors. Hence, one can say that all clusters are identical. Here we will have more than one level of decision making. So in every cluster, the cluster head should decide on the required action in its cluster and in the next level there may be the sink node or one other layer which also has been clustered and its head should make a decision about the transmission action. [21]

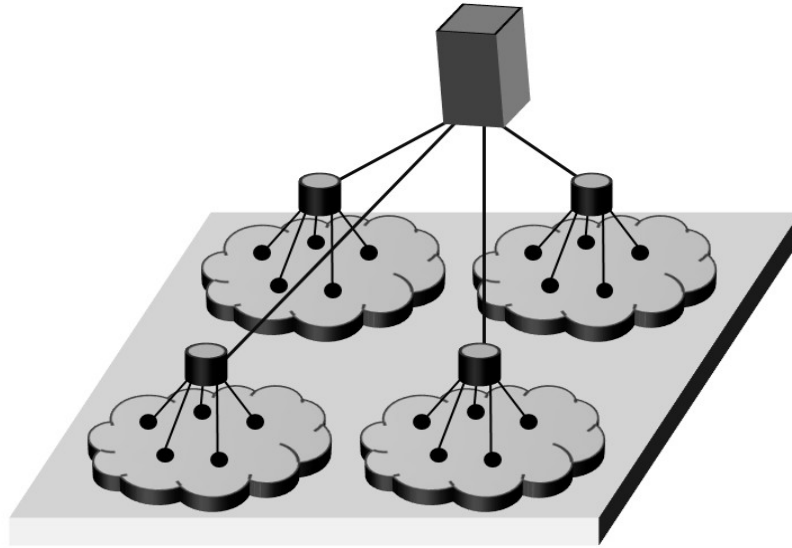


Figure 9: A 3D view of an uniform clustering with two decision levels

In Figure 9 a network consisting of four uniform clusters is shown, which in each of them, there are five sensors nodes. Consider how the level of decision making is different in this kind of network.

Figure 10 illustrates a comparison between a non-clustering network and a clustering network with different level of decision. [21]

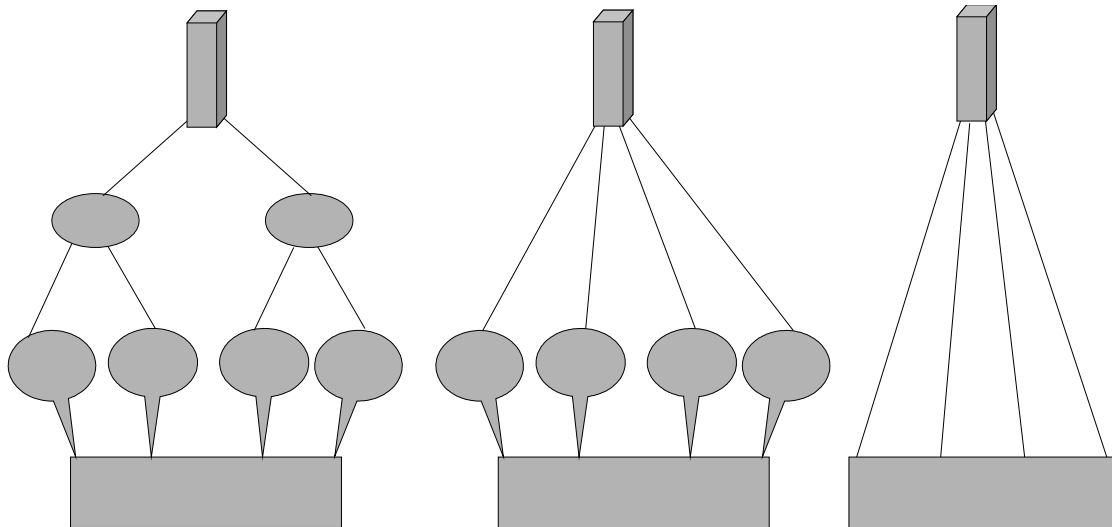


Figure 10: From left to right, A view of: A- A non-clustering network; B- A uniform clustering with two levels of information fusion; C- A uniform clustering with three levels of information fusion.

4.7.3.2 non-Uniform Clustering

In non-uniform clustering, the clusters are different from each other. So we can expect to find different number of sensors in every cluster. This kind of clustering has its own considerations and issues, but however, it would be the closest type to the models used in real world applications. A general view of a non-uniform clustering is shown below.

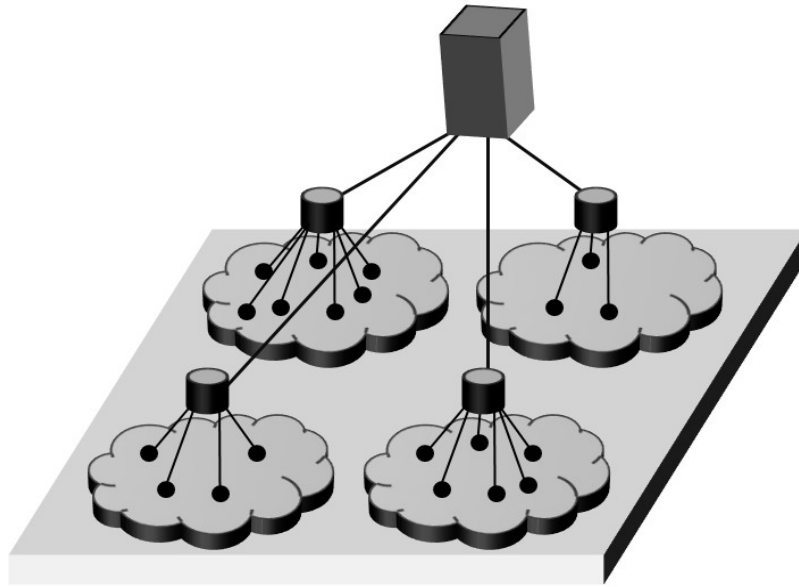


Figure 11: An example of non-uniform clustering with total number of 20 sensors

4.8 Network topologies

Smart dust networks basically point to large scale wireless sensor networks which contain a large number of low power MEMS devices, called motes; which will automatically organize themselves in a network using the concept of self-organizing maps (SOM) discussed in 3.3. These small and low power consuming computational elements have different kinds of sensors which allows them to detect any kind of crucial events in their environment.

There are different topologies for implementing mote networks. They are basically classified by their power consumption and also the time it takes for a mote to report the event in its surrounding to the base station. So it is about efficiency in the speed or the consuming power.

These networks are also divided in two main groups in terms of the way they send their messages to the base station. In this category they are divided into single-hop or multi-hop message passing policies.

Single-hop communication is so efficient in terms of speed in sending the messages to the base station. On the other hand, multi-hop communication consumes less power, is more secure and it also avoids traditional signal propagation. By decreasing the number of hops in multi-hop communications, we can expect to have less power consumption.[2]

We have considered three possible implementation based on our designed protocol. Below you can see more details about these topologies along with their advantages and disadvantages.

4.8.1 Dynamic networks

In a dynamic topology we will have an area which has been spread accidentally by several motes. The idea here is to assign an ID to each mote dynamically.

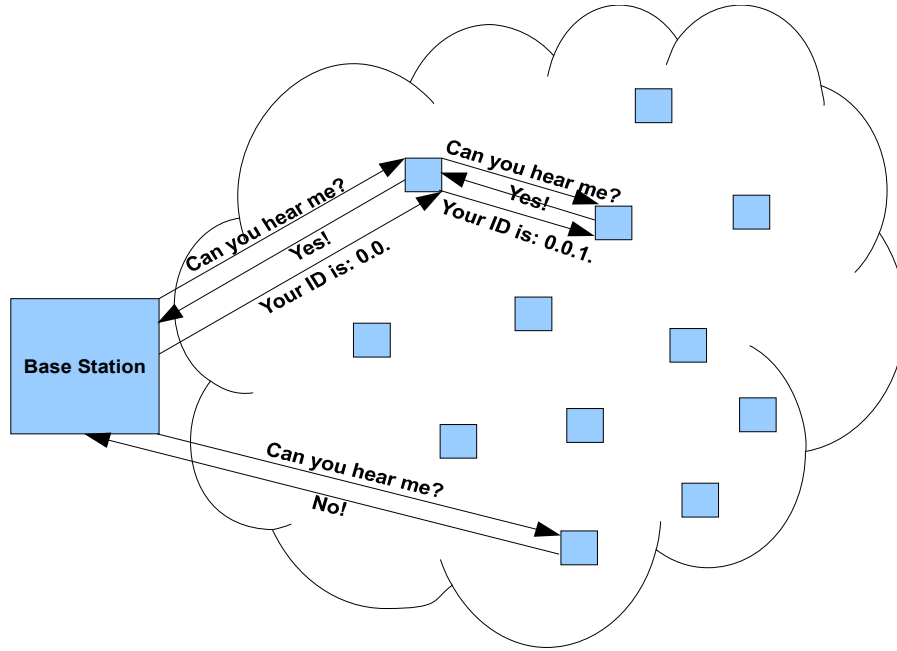


Figure 12: A simple presentation of Dynamic networks and the way of ID assignment

4.8.1.1 Basic Definitions in Dynamic Network Protocol

The idea in dynamic network topology is to assign an ID to each mote dynamically. It means that we consider a distance as the proper distance for two motes to become friend (we will name this distance the 'friendly distance' from now on). A process of giving IDs will be started in a cycle from base station.

In this topology it is supposed that every mote has its position in the area based on its geographical length and width that we named them traditionally as x and y respectively. The source for counting this position is the base station which has the position of. This way every mote is able to measure its distance from the base station using:

$$d = \sqrt{X^2 + Y^2} \quad (1)$$

4.8.1.2 Data propagation Algorithm in Dynamic Network Protocol

This cycle starts by a message from base station to the very first mote in the area close to base station. This mote can be saved in base station as the first registered mote. Base station will send a message to verify that this mote is placed in its friendly distance. If mote find itself-in the friendly distance from base station, it will answer to base station message and verifies that it is able to communicate properly. Then the base station will assign the first ID to this mote, otherwise it will go to the next mote in its list and repeat the same process. When the first mote found itself-in the friendly distance of base station, it considers itself-as one of the direct children of the base station and introduce itself-to the network by its assigned ID from base station. Then this mote also starts the same cycle as it has experienced from the base station with the only difference that this time it plays the role of a base station and tries to find its children dynamically and assign an ID to each and every one of them. This ID is an extension of its own ID which shows to others that who belongs to which mote.

Every mote will start the same process as soon as it receives an ID from its parent and this process will be continued until all the motes in the network has an ID and know their parents and children.

The data flow is always in a sub area of motes which are all connected in this way. It is expected that in this

way every mote knows a low cost path in terms of energy consumption toward the base station.

4.8.1.3 Advantages

This method is considered to be a multi-hop communication method and have a lower power consumption. It is also more secure by definition. Moreover, as it is mentioned before, this kind of network cancels the signal propagation effects in long distance in wireless sensor networks.

This topology provides very high flexibility and reliability in the network and it is presumed to be the most efficient way of setting up a network. Beside that if one of the motes stop functioning or go out of order for any reason such as being totally destroyed or switching to harvesting mode to collect more power. This is possible since there is a sub-area of connected motes which gives a different low cost choice to the mote.

4.8.1.4 Disadvantages

Implementation of this topology is so complex and time consuming. It is not known for sure that this level of flexibility is possible in Erlang.

4.8.2 Semi Dynamic networks

In this topology, IDs will be assigned along with an 'add_mote' command from the customer. After that they will be organized in a proper order by the base station. Here we don't assign the IDs dynamically but we organize them in a dynamic manner. So finding upper or lower level motes will be done by motes and base station at the set up time. Here we probably do not need to assign any position to motes.

4.8.3 Static networks

In this method every mote has its ID in adding time and its position in the network is already known. So there is no necessity in giving an X and Y for the positions coordinates to every mote. Parents and children of the mote are already known and the position in the network is fixed.



Figure 13: A simple view of a Static Network; Fixed place for every mote and base station can be seen clearly.

4.8.3.1 Advantages

Implementing this kind of topologies is not that much complex. The Setup time is shorter and if all the motes work properly, this network will be faster in comparison to other topologies. This means a mote far away from the base station can send its messages to the base station in shorter time.

4.8.3.2 Disadvantages

Static networks are not very flexible and reliable. They are not trusted in the sense that all the links in the network are fixed and if one of the motes, which is in the vital node, go out of order, it may happen that a

whole branch of nodes in the fixed tree lose the possibility of connecting to the base station. The reason for this problem is that static networks do not have the ability of dynamic organizing. Therefore, as soon as it loses one vital connection, the whole network becomes practically useless or it is not able to operate a part of an area. So they need to have very good strategies to make it possible for the network to survive even with losing one or even several of its nodes.

One way can be using single hop communication which increases the power consumption of the node but also improves the network speed and solves the mentioned problem.

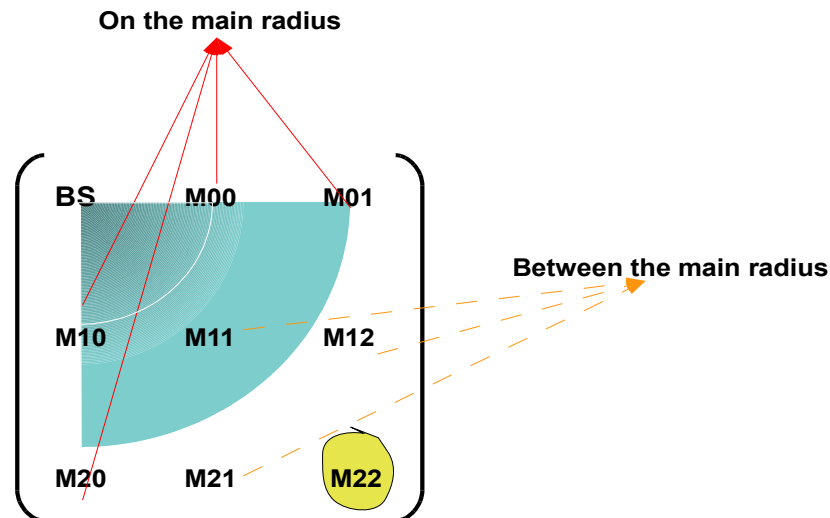


Figure 14: A presentation of distance measurement in a static network

4.9 Data Propagation algorithms in Smart Dust

4.9.1 Geographical distance issues

So one question that comes to mind is why having a wireless sensor network which runs with a low power consumption is important and where is vital to deploy this type of networks. One simple answer which uncover itself quickly, is supervising process on an environment which is hard to be accessed by human, for instance remote jungles in Australia or Mars or environments which are dangerous for us to be in, for example an area which is polluted by nuclear explosion or similar. [37]

As an example of a case where smart dust can be used for local detection and fast communication to the base station in the presence of a crucial event, imagine that millions of tiny disposable wireless sensors are sprinkled over a forest from an aircraft. Every sensor can monitor the temperature of a small geographical area. Quickly they will set up a dynamic and efficient communication network among themselves, distributing the tasks of monitoring the region in a specific network topology. They are all alarmed to signal any possible chance of fire to the network and in the end to the authorities to avoid large damages. [37]

There are some special aspects of these networks should be taken into considerations; Critical case such as very low and poor battery power, no synchrony and low computation abilities. There should network protocols to deal with cases such as failure of an individual or a group of sensors while maintaining low power requirements. Network scalability is another important issue to change the topology, node density and the size of the network. Because nodes might be destroyed, die or even be added to the network, network topology should adapt accordingly over time. [37]

4.9.2 Jamming

Jamming in word means to extract an object into a defined or limited area. Here we use this term in radio science world where a communication line/system will be disrupted by radio signal transmission which is the result of signal to noise ratio degradation.

If a transmitter sends more signals without noticing that the frequency is already in use, an unintentional jamming will occur. This normally happens on frequency lines with high percentage of usage. Unintentional jamming can also be the result of the fact that transmitter has not been able to hear stations using the frequency.

This subject can be of concern for smart dust networks as well, though we close this issue here by a short introduction to that and introducing it as a possible point of concern and research interest for smart dust networks.

5 STUDY ON SIMULATORS FOR SMART DUST NETWORKS

5.1 Introduction

This chapter is mainly focused on available software and simulators for network simulation. There are a variety of simulators for simulating different kinds of networks with different purposes. One of the basic steps for implementing this project was finding a suitable simulator that fits all of our requirements.

The aim is to know which simulators are better for simulating a basic network for smart dust. A research was done to understand basically what kinds of simulators have been used in different research projects for implementing a distributed sensor network (DSN). The result of this research is given below.

5.1 Simulators for Sensor Networks

There are already simulators out in industrial or research level that are able to simulate different aspects of a sensor network and help to overcome the challenges regarding this type of networks.

Every simulator has shown to be proper tool for simulating one or more characteristics of sensor networks. Below some of these tools is introduced that have been already used in other research works on sensor networks around the world.

5.1.1 GloMoSim

GloMoSim or “GLobal MOBILE Information system SIMulator”. The tool has some pros and cons. It has the same issue as NS-2 in regard to energy consumption modeling. A good feature is that it supports ad-hoc networks with lots of protocols. However it doesn't have routing protocols for sensor network. The best advantage of this simulator is that it is a free simulator which its last version and the manual guide have been downloaded for later use. [2]

5.1.2 System C

SystemC is a programming language which has designed based on C++. [8]

5.1.3 Tiny OS

TinyOS or Tiny Micro-threading Operating System, is an open-source operating system which has been developed by university of California Berkeley. This operating system has generally designed for simulating low-power wireless devices like wireless embedded sensor networks.

TinyOS is very useful for devices based on microcontrollers. It is specifically popular for its good performance on devices that have sensors, networking capabilities or both. TinyOS is designed for devices that have to meet very sensitive resource constrained like devices that have to meet very hard constraint for power like smart dust which needs low power networks.

TinyOS has some problems though; it has a different programming style and it takes time for programmers to learn and get use to it. It's also not so efficient in case of intensive computation applications. [15]

5.1.4 NS-2

USC has used NS-2 which is a discrete event simulator and network animator that supports both wired and

wireless applications. This simulator is widely used for network simulations and has lots of resources for learning.

The other thing is that based on the comments given by users, learning and working with this simulator is time consuming and it does not seem to be a proper choice of simulator for a master thesis work. But I think the worst thing about NS-2 which makes it inadequate for this project is that it cannot model the energy usage in the network which is an important factor for this project. [7]

With attention to the history of smart dust and DSN we can see that the main problem in these networks is the energy consumption. Researchers are trying to increase the mote's lifetime and decrease the size by degrading the energy consumption of the motes. As we know, most of the energy consumed is due to transmitting and receiving data. So we need to know the energy usage of the network. This can be used for implementing a basic smart dust network, but it doesn't seem to be useful for improving algorithms.

The last but not the least problem with this simulator is the limitation of number of motes in simulation; we cannot increase the number of motes more than 500 in this simulator. [7]

OSU has used JavaSim for sensor network simulation; but its main focus is on wired networks. However it is a user friendly simulator. [7]

5.1.5 SensorSim

SensorSim is an extension to NS-2. It provides battery models, radio propagation models and sensor channel models. It is believed that SensorSim is the closest simulator to sensor networks. Since this is a new tool, we should not expect to get lots of documents and tutorials about it. The other disadvantage is that it is a small scale simulator and it cannot support more than one base station. [2] [7]

5.1.6 JavaSim

JavaSim is a free discrete event, object-oriented simulation package that is available since 1997. It is written in Java and it comes with a set of examples and test routines. The original JavaSim was based on HASE++, which was a C++ simulation library.

5.1.7 Erlang

Erlang has been developed by Ericsson AB and usually known as ERicsson LANGUage which describes the name. It also has been said that this name points to also the name of a Danish mathematician named "Agner Krarup Erlang". Ericsson has the highest score for using Erlang. AXD301, which is one of Ericsson's products and is released in 1998 was contained over a million lines of Erlang. [5] [12]

The first version of Erlang which was implemented in Prolog is developed by Joe Armstrong from Ericsson. [12] You can have a more detailed description on this simulator in section 6.1.

5.1.8 OPNET

OPNET which is used by OPNET Inc. is a commercial network simulator with a fairly good stability which implements 3 protocols for ad-hoc networks. As a down side, it does not have energy usage modeling and enough protocols for wireless networks. [7]

5.1.9 Other simulators

"Magic Weaver", "SensorSimII", "DASSF" (Dartmouth scalable simulator framework) by Dartmouth University or "UIUC lager scale sensor network simulator" by UIUC are examples of other simulators developed. [7]

5.2 Erlang

After searching between possible simulators for simulation of a distributed sensor network, we decided to use Erlang. Erlang is a functional programming language which supports very good level of concurrency. A

functional programming language which has a good degree of concurrency and high reliability. [11]

5.2.1 What is Erlang?

As official website of Erlang defines Erlang as a concurrent general-purpose programming language which supports run-time environment and provide the programmer with fault tolerant characteristic. [10]

The basis of Erlang is basically on message passing which makes it perfect for our application. For this reason it is a good idea to be familiar with its actor model.

5.2.2 Erlang History

Erlang programming language has been designed and developed by Joe Armstrong for Ericsson company first time in 1983. From then, this language has found much more exciting usages around the world and many programmers adores this programming language for its specific features and use it in much different subjects than it was designed in the first place.

5.2.3 Erlang Actor model

We said earlier in this document that Erlang is a language which is based on message passing. To be able to understand how this message passing mechanism work in Erlang, it is beneficial to know more about the actor model of Erlang.

In simple words, imagine the simulator as large number of people who are sitting alone in a dark room waiting to receive a message. The only action that each person does, is to reply to messages in a proper way and maybe ignore spam messages. Then a communication method is specified and these people are able communicate with each other through this method; let's say writing letters. We can also assign each person to reply to a specific message and ignore the rest. [11]

This is exactly what happens in Erlang. Erlang also uses this model to be able to perform lots of tasks at the same time. Every person in our example would resemble a separate process in Erlang.

This is the case in our project as well. We need some nodes that can only send and receive messages. (if sensing, sleeping and harvesting processes are ignored). But we should remember that the most power hungry action in the network is data transmission which is also our main focus in this thesis work.

5.2.4 Erlang download, installation and other

To start programming in Erlang, we need to have the software. Fortunately in 1993 Ericsson released Erlang as an open source language. So different versions of this software can be downloaded from its official website [10] and also from SUNET [38] which is the Swedish University Network.

We downloaded version 5.8.3 of Erlang and used it all the way during the Erlang programming of this thesis. For Erlang editor notepad++ was chosen for its good support on Erlang formats. In this project Erlang is installed and used in a Windows machine. However, Erlang has its own virtual machine and supports very unique features like hot load coding and powerful concurrent application. [10]

In 1993, Ericsson added the distributed Erlang to the software to make it more reliable for distributed applications. With this feature, it will be possible to run two process at the same time on two different Erlang server and these processes can communicate together online and effect each other work. This is also the case when we run two processes on two different computers.

5.2.5 Erlang Applications

Erlang/OTP has developed for Ericsson specific applications at first but over years it has shown itself to research and industrial market as an undoubtedly one of the most powerful tools available. Having this in mind, we must bring this point into consideration that any type of technology or tool has its own limitations that also limits its applications. Erlang has shown the best results and most interesting usage under distribution subjects. It perfectly covers the need of having a software to write arbitrarily distributed over multiple instances. Using Erlang, it is not even needed to change the programming paradigm for any

instance case.[16]

On the other hand, however Erlang is equipped with a highly optimized VM, still is not the best choice when it comes to execute a time critical processes. So if you are working with time critical applications, you may want to erase Erlang from your list.[16]

Erlang also has a great usage in back-end systems such as databases, queuing messages and abstracting the storage. It is probably one of the best choices possible when it comes to write a reliable software too. [16]

Having all these in mind, one can guess if Erlang is a proper choice just by looking into the nature of the problem which she/he is faced with. Is the problem under any distribution or reliability subjects? Or is it facing with redundancy or scalability? If yes, then Erlang can be a good choice to start with. This is a good start of the way toward choosing a proper tool, though it is not enough. Here, one must look at the nature of Erlang too, considering its weaknesses and strengths and decides if this language can cover what it takes to solve or simulate an issue.[16]

5.2.6 Concurrency in Erlang

“The world 'is' parallel – we are parallel”. [39] This is a quot from Joe Armstrong, designer and developer of Erlang. So if we want to simulate a real application we have to implement it in concurrent way to be able to understand the real effect in the real world.

Erlang is designed for such applications and therefore makes it a very good choice for our project. The other feature which makes it qualify for our application is its high speed in message passing.[9]

5.2.7 Distributed Programming in Erlang

Distributed Erlang is one of the most exciting features and a good measure of its power. So how distributed Erlang does work and what kind of problems can be solved by this important feature?

Imagine having a really large program that is running slowly on one machine and the speed is just disastrous. What can be done is taking benefit of two main Erlang features, distributed Erlang and its message passing nature. So now, you are able to break your program into smaller parts and accommodate every small broken piece if the large program in a different machine. Using the distributed Erlang, now you can run all small program on different machines parallel and they can communicate through message passing with each other. The result would be the same however you have managed to increase the performance considerably.[9]

Basically a system which is using this feature of Erlang consists of number of nodes. Nodes are basically are Erlang run-time systems which communicating with each other. The message passing actions mostly happens between these Erlang runtime elements. The distribution mechanism is implemented using TCP/IP sockets. [10]

Also if there is the probability of fault and we want a safe complete run of our program, then it is possible to make a fault-tolerant system by distributed Erlang. Based on this, we make the system to run on several machines. If one machine fails to finish the run of the program, program can continue running on another machine. So we have a good reliability by using distributed Erlang in this case. [9]

One of the important advantages of distributed Erlang is scalability. When we try to scale up an application, we should always consider the capacity of the system to see if our machine can stand the change. It adds the cost and effort for an application scale up. But it is obvious that if we keep expanding our application, it comes a time that one machine (however the most powerful one) does not have the capacity for this purpose. Here we can again use the distributed Erlang to help us to divide the capacity on more than one machine So we will be able to add more machines and in result more capacity without the need to make basic and lots of changes to our application.[9]

Beside all of these advantages, we should accept that there are a large number of applications which are distributed inherently. For example the applications that are multi-user and in which different users are using the application but they are scattered in a vast geographical area. For instance if we want to write a multi-user game which different users in different geographical places can access to that game, then we

faced a problem which is inherently distributed. The same is for a chatting system.[9]

In a distributed system with several Erlang nodes, there may be a need to control applications in a distributed manner. If the node, where a certain application is running, goes down, the application should be restarted at another node. Such an application is called a distributed application. Note that it is the control of the application which is distributed, all applications can of course be distributed in the sense that they, for example, use services on other nodes.

Because a distributed application may move between nodes, some addressing mechanism is required to ensure that it can be addressed by other applications, regardless on which node it currently executes.[10]

There are two main models of distribution which are discussed in the following sections.

5.2.7.1 Distributed Erlang

Distributed Erlang develop a way of programming for applications that run on a set of tightly coupled computers. Here programs make to run on Erlang nodes. A process can be spawned on any node, and everything about the message passing and error handling primitives can be managed like it happens with concurrent Erlang.

In distributed Erlang applications every node can run any application on the other node, so there should be high degree of trust between nodes and we need to run such applications in a trusted environment. Generally distributed Erlang applications will be run on “clusters on the same LAN and behind a firewall” but also they can run in an open network.[9]

5.2.7.2 Socket-based distribution

But what if we need a distributed application and we have an untrustworthy environment. We can write distributed applications using TCP/IP sockets, and this one can be run in an untrustworthy environment. The programming model is not as powerful as the one we use in distributed Erlang but it offers a good degree of security more than distributed Erlang.[9]

6 ALGORITHMS AND POLICIES

6.1 Introduction

In this chapter we will explain our suggested algorithm using figures and more. Later in this chapter, every part of this algorithm will be explained in detailed.

6.2 A general view on our algorithm in Erlang

Most of message control between mote network and customer processes will be handled by base station. Here we will mainly focus on data types, link types between different processes and a general view of the system. As it can be seen in Figure 15 system consists of a network of motes, a base station and a client.

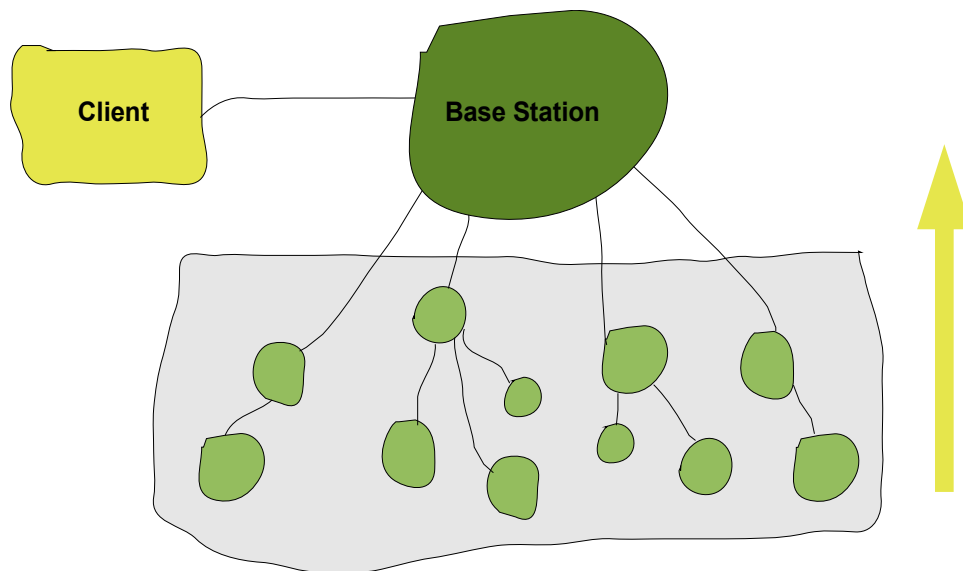


Figure 15: A simple view of the system and direction of data flow

6.3 Power management policy

The idea in this project is minimizing the power consumption due to transmission as much as possible. As Pister and Kahn has reported for a mote in a cubic millimeter volume, the maximum stored energy of 1 joule is possible for example due to solar cells. So for having the mote active as much as possible, their energy consumption during a day cannot exceed from micro joule level. [5]

An excellent power management would be essential to limit the energy consumption of a mote and increase the mote and also network lifetime. Power management strategy in this project mainly concentrates on software point of view and ignores the hardware solutions. It assumes that this kind of solutions will be applicable in all kinds of hardware.

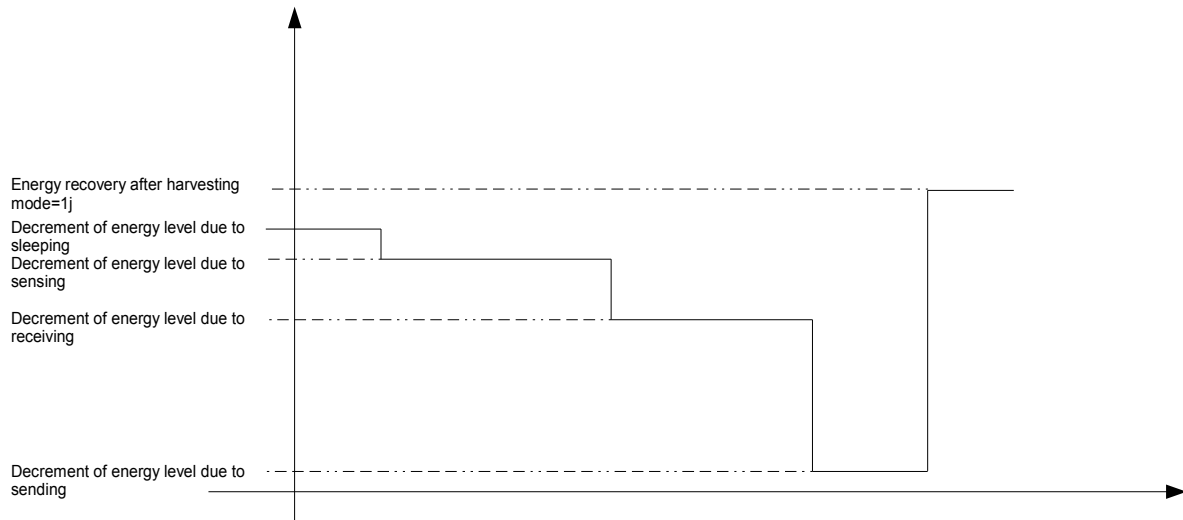


Figure 16: A simple presentation of different energy usage levels for a mote

For limiting the power consumption of a mote, different power consumption levels has been considered for every mote. Based on this energy management strategy every mote has different activity mode and every of these modes have specific power level. By using this strategy motes won't stay in fully active mode all the time. The different energy levels, shown in Figure 16 are described in more details below.

6.3.1 Mote behavior algorithm

Mote behavior has been implemented based on the concept of FSM, shown in Figure 17. Motes behaves completely as we discussed previously. It is clear from this figure that we have not considered the harvesting mode yet. The problem with this was that we did not consider the power consumption in this FSM. So it definitely will need to be implemented while we are considering the harvesting mode too.

The other issue here is that, this may become necessary that we need to bring this module somehow inside the base station server module. So it was decided to concentrate on the base station server module and wait to see if this change is needed.

Now lets have a more detailed view on the implementation in Erlang.

Since the network is implemented in small scale, "Orddict" is used as the data type for every mote. This type is introduced to the network by base station server module when it adds some mote to the network by initiating an object.

The relation between motes has been considered to be of monitor type, as they have to survive when the other mote is dead. But the relation between motes and base station is of link type, as they should not be able to work without base station. There are some uncertainties about the types of links that have to be investigated more. However this link is important when we want to add lots of motes to the network in server module but not when we are implementing an individual mote behavior.

So, as it can be seen in Figure 17. a mote is initiated in sleeping mode and based on different inputs which is impose to the mote in different condition, it reacts accordingly, as it is discussed in the theoretical part.

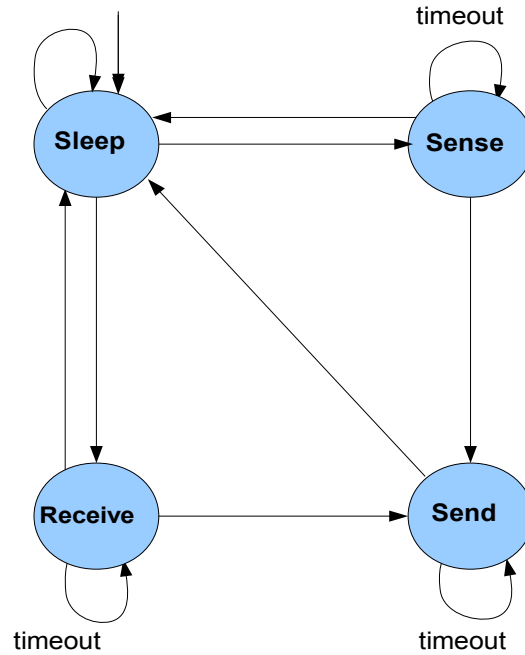


Figure 17: The FSM that describing mote's behavior

Since there are not any available experimental results, exact numbers for timeout and also different levels of energy has been ignored and are just handled through passing proper messages by motes. With this approach we have more reliability and flexibility in our code, which makes it possible to easily be replaced by another text, number or even related formula.

6.3.1.1 Sleeping Mode

Every mote by default is in sleeping mode which has been considered as the lowest power consuming active mode. In this mode, mote is alive and is listening to the network but do not use much power. If we consider that the highest possible saved energy is about 1 joule, we can assume that a mote will use 1% of its saved energy in this mode. Every time after switching to other modes with higher power activities, motes intend to return to sleeping mode. Even if they don't receive any more serious danger signal in a specific timeout, they will get back to sleeping mode.

Because we do not know exactly what would be the exact energy usage in this mode, we named it in our code as "*Energy usage of class IV*". So level IV is the lowest possible energy usage when a mote is alive.

6.3.1.2 Sensing Mode

If sensors in the mote detect any special changes in the environment where they are monitoring, there is a chance that a possible error (or in simple words, a danger) in some part of the related region (the position that mote is placed) has occur. So a signal with a power above the safety threshold will be imposed as an attention signal to the mote.

Hence, the mote has to go to a more sensitive mode and start to listen to its sensor more carefully. If the mote receive more alarm signals from its sensor, the danger level considered to be serious and mote will go to sending mode to be able to inform its parent about the strong sensed signal. Otherwise if mote does not receive more signals in specific timeout, the first alarm signal considered to be a noise or a false alarm. In this case, the mote after passing a timeout waiting for more attention signal, will immediately go back to sleeping mode. The energy level in this mode is the second lowest power consumption and we call it "*Energy usage of class III*".

6.3.1.3 Receiving Mode

In receiving mode, the mote consumes higher energy than sensing mode. We name it "*Energy usage of class II*". This mode will happen if one of the children of a mote, stays long enough in sensing mode which makes that mote eligible to go to sending mode and send the sensed signal to its parent. In this case the parent mote should go to receive mode to be able to receive the message from its child. The parent mote will stay in receiving mode as long as it receives enough messages from its child in specific timeout.

If the mote receives specific number of attention messages from its child, danger level will be considered high, and the mote goes to the sensing mode. This process will be repeated until danger signal reaches to the base station and proper action will be decided by base station.

If after passing the timeout, the mote does not receive more signal from its child, the previous signal will be regarded as noise and therefore ignored. After passing timeout and not receiving any more signal, parent mote will immediately go back to sleeping mode to avoid using more energy in receiving mode.

6.3.1.4 Sending Mode

Sending mode suppose to be the most energy consuming action in the network which deplete the mote's energy soon. The strategy which has been used in this project resists going to sending mode as much as possible and try to do this just when it's vital to be done. Ignoring to go to the sending at required time will have risky consequences for the network. The degree of this risk, will be different based on the application that we want from our network. A mote should go to the sending mode in two conditions described below:

6.3.1.4.1 Sending after Sensing

There is a possibility that the mote switches itself-to the sending mode. As it is discussed before, when the number of signals, which their strengths are above the safety level, reaches to a specific threshold number, the mote has to go to the sending mode.

When a mote triggers itself-and go to the sending mode, it means the area of risk, for example the place that the fire has started, is in the same position as the mote.

6.3.1.4.2 Sending after Receiving

There is another possibility in the network which causes a mote to go to the sending mode. It will happen when a parent mote, receives the specific number of alarm signals from its children (discussed in 6.3.1.3).

In this case for finding the source of the danger, we should keep a record of the sending motes and the position of the mote where the place of interest is at the top of the list.

A mote will go to the sending mode by one of the above processes. After sending the message to the upper level in the network, mote will return to the sleeping mode as fast as possible to avoid wasting energy in sending mode.

6.3.1.5 Harvesting Mode

After a chain of energy consuming activities consist of sleeping, sensing, receiving and sending, the mote will be depleted gradually. If a mote does not have enough energy to be able to go to one of the active levels (i.e. sensing, receiving and sending modes) from sleeping mode (which is counted to be a semi active state), is considered to be out of order and in need of energy. In this case the mote will send a message to the network saying that it is out of order and needs to go the harvesting mode.

So based on the network structure, it is possibly unavoidable for network to continue to its work without re organizing itself. After saving enough energy, the mote will be able to return to its place in the network and the network is also able to go back to its initial organization order.

6.3.2 An example of energy usage by motes

The diagram in Figure 5 shows a possible cycle of energy usage by a mote. As you can see, a mote has basically 1 joule saved energy. As it is discussed previously, a mote is initially in sleeping mode which is the

lowest energy consumer between different possible active or semi active modes for a mote.

The mote will stay in sleeping mode until it receives an attention signal from its children or sense some changes in its proximity. In our example after being sleep for 8 hours, the mote senses a signal above the safety threshold and goes to the sensing mode, waiting to sense more signals. After a timeout of 15 minutes, the mote does not sense any more risky signals. It means that probably there is not any noticeable danger around. So, the mote consider this signal as a false alarm and returns to sleeping mode while keep listening to the network. Assumed that the purpose of this network is to detect fire in a forest. These false alarms, for example, may happen if the temperature during especial hours of the day rises so high and comes back to normal again after a while. Practically there is no fire, but the sensors detect a high risk signal from the environment.

Going back to the example, the mote is back on sleeping mode and will stay in this mode for 5 more hours. Then it receives a serious signal from its child and goes to receiving mode. This time it is a serious alarm and the signal is passed through the network to the base station. After the base station receives the threat signal, knowing the location where the alarms started, it can inform the authorities to take the necessary actions; which in this case will be notifying the fire department.

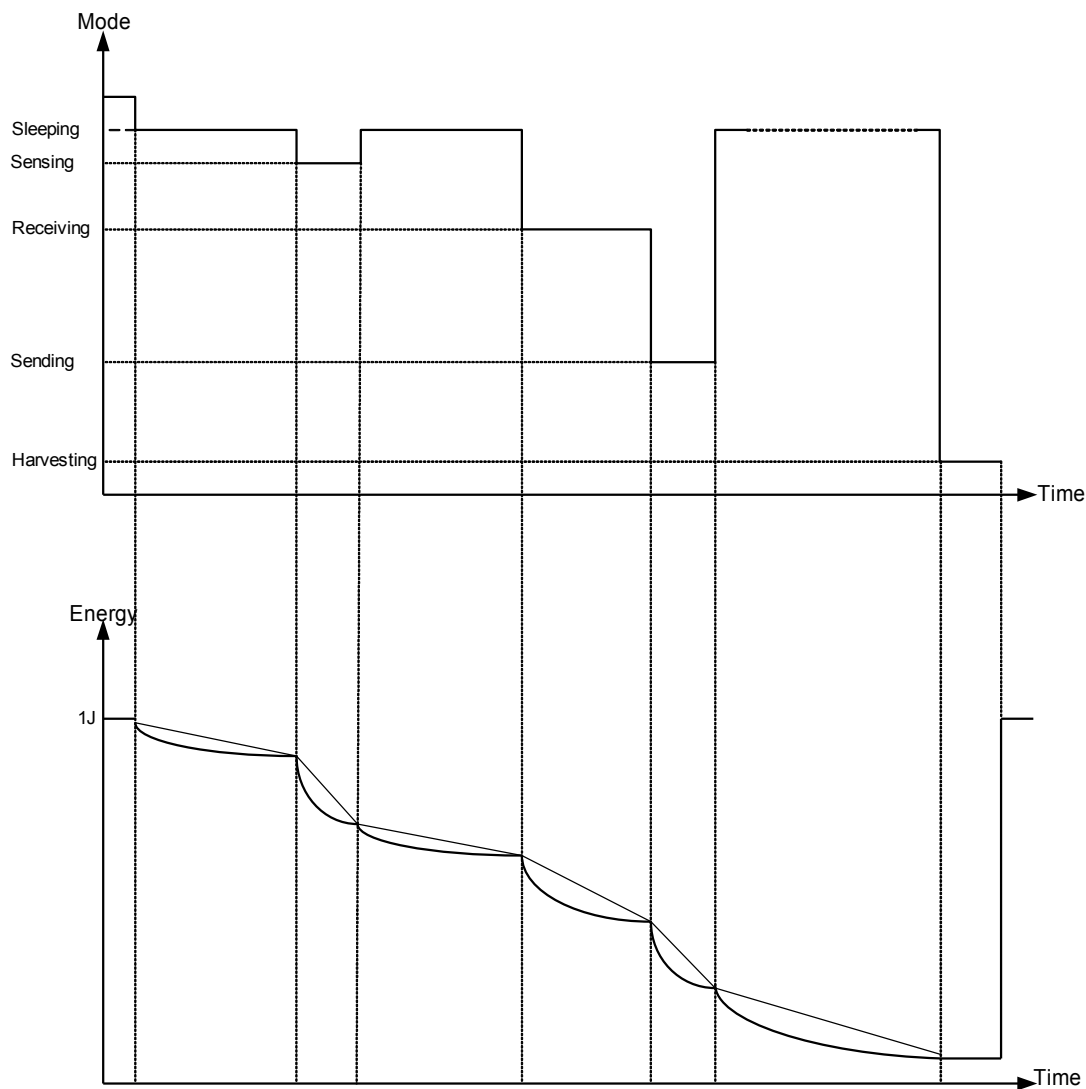


Figure 18: An example which shows the change of energy levels in a mote for different modes

6.3.3 Policies in case of possible clashing

Clashes in this module is highly possible. It means there is a possibility that a mote is in sensing mode and at the same time has a received message in its inbox. It is decided that if the mote is in sensing mode and receives an imposed alarm signal from its children, it will go to receiving mode as it is obvious that another mote has sensed strong signal above the safety threshold which makes it inaccessible to its parent do the same thing. This condition informs that a serious danger is sensed and it is essential for the parent mote (which is now in sensing mode) to go to the sending mode.

If the opposite scenario happens, meaning that the mote is in receiving mode and sensors also detect something in environment, the same decision is taken, i.e. the mote will ignore the detected signal from its sensor and stays in receiving mode.

Based on this two parts, receiving has more priority to sensing. This is also the case for sending mode. When a mote is in sending mode, it ignores all the incoming signals from its sensors and also from other motes. In harvesting mode, mote is not active, so all incoming signals will be automatically ignored.

7 ERLANG SIMULATION RESULTS

7.1 Introduction

The coding consists of three main files:

1. Mote.cc
2. Mote_main.cc
3. Mote_gen.cc

In “mote.cc”, the definition of a mote is done. Also, the various functions of a mote is defined, such as harvesting, sensing, sending, receiving and sleeping. These functions use command (2) to show the state of a function.

cout << sc_time_stamp() << . (2)

The “mote_main.cc” file consists of instantiation of a number of motes. The definition of a mote is already performed in the file “mote.cc”. However, the motes need to be instantiated based on this definition. This exact task is done in this file. Nine motes (Mote 1 to mote 9) are instantiated, each with the details of the immediate neighborhood mote. This way, each mote knows its immediate neighbor and thus can transfer information to its neighbor.

The simulation of the various motes in tandem is performed in the file “Mote_gen.cc”. The simulation is started in the function calls “Mote_gen::gen()” and “Mote_gen::output”. Basically, to start the simulation we need to set the sensor inputs (sensor_input) of the nine motes instantiated in the “mote_main.cc” file.

7.2 Mote.cc functionality

The functionality of a mote is mimicked by using the functions: Harvesting, Sensing, Sending, Receiving and Sleeping. The energy of each mote is defined using an “energy” variable. All the nine motes are running in parallel and independent of each other. The simulation functionality in the file “Mote_gen.cc” decides which mote will be triggered first.

Looking at the individual modes of a mote defined in this file:

- Initially, every mote remains in the sleep mode. It breaks out of the sleep mode, when the sensitive signals “higher_threshold” or “received_signal” is triggered. Based on this triggering, the mote goes into “sensing” mode or “receiving mode” respectively.
- The harvesting mode of a mote just increases the energy of a mote and suspends all its other operations.
- The sensing mode of a mote initiates the sending mode of the same mote.
- In the sending mode, the mote tries to send information to the neighborhood mote, but if the neighborhood mote is in harvesting mode, then it sends to the neighbor of the neighborhood mote, or to the base station.
- In the receiving mode, the receiving mote just displays the message that it has received from another mote.

7.3 mote_main.cc functionality

As stated before, the “mote_main.cc” file consists of the instantiation of a number of motes. The total number of motes used in this part is 9. All the motes have the same structure and functions as defined in the file “mote.cc”.

Once the instantiation is performed for all the 9 motes, each one of them waits for the suitable “higher_threshold” or the “receive_threshold” signals to start their respective works.

7.4 mote_gen.cc functionality

The simulation of the project is handled by this file. In this file, there are two sub functions:

1. Mote_gen::output()
2. Mote_gen::gen()

The “mote_gen::gen()” function starts the simulation by generating random numbers to trigger a specific mote and then calls the “mote_gen::output()” function. The “mote_gen::output()” in turn sets the “sensor_input” signal of a particular mote, based on the random number and that particular mote comes out of its sleep mode and gets into sensing mode.

8 CONCLUSIONS

Wireless Sensor Networks refers to types of networks which shape by a number of sensors or sensing elements. The primary use of this type of network has been to be spread in an environment to sense and follow one or several characteristics of that environment. This continues monitoring of a limited or unlimited area, has made a large and interesting research area.

Smart Dust networks in the other hand are also placed in the category of ad-hoc network as well as under distributed networks classification.

Smart Dust basically points to large scale wireless sensor networks which contain a large number of low power and small computational elements so called as motes. Motes can be used in applications which demands that they automatically organize themselves in a network. This can be gained by using the concept of self-organizing maps which has been discussed in detail earlier in this thesis work. They are able by assumption to be able to overcome the computational problems in the environment they serve in, they can be capable of providing their own power by harvesting energy from environment like solar energy, they of course can and must be able to perform a vast range of communicating actions with other members of the network and much more. This possible characteristic among other interesting possibilities which this type of networks provide, has made a vast range of possible research areas as well as a vast and interesting range of applications which can make human's life on earth much easier and much smarter. While Smart Dust network belongs to the class of wireless sensor networks, this powerful result of human brain shares the same level of interest for researchers as well as for a group of Master and PhD students at Electronic System department at Linköping University. So a large scaled project was defined and this thesis concerns a really small part of the whole thesis.

Considering the above scenario, if we have a self-organizing smart dust network, when a mote settles in the network, it starts a series of activities such as creating a connection with other motes, sending and receiving and many more activities based on their application, their technology and their design. But what they can limit the power of these small smart elements is their power consumption. Pre-study phase of this work suggests that motes normally live in high activity mode whether it is needed or not and also they waste considerable portion of their saved energy by communicating through the network.

We started this thesis work by finding a solution to power consumption problem in smart dust networks. We have considered many proposals to be able to solve this problem partly. At the end, it was decided to focus on every mote behavior and try to improve mote's power consumption attitude.

Studying the results of other researches that was done in this area, it was clear that most of the motes in those studies have two states of working, means sending and receiving and most specifically by sending. It was a question why a mote shall spend all its life time in these two states while the results of some other studies showed that a mote spend most of its power in sending and receiving process.

All these guided to design an algorithm that concerns basically mote behavior through their life time. This algorithm suggests that more than two states of activity shall be added to a mote life time. All the added states shall need no or minimum level of energy consumption but must be used as a medium activity while sending and receiving are not a necessary actions to be taken by a mote. Adding these states must take a mote out of power expensive modes of sending and receiving. This algorithm shall not let a mote to enter to high priced modes if the computational circle of the algorithm has not recognize the situation as critical. By this primary explanation in mind, two other modes called sleeping and sensing were added to the life cycle of a mote. This algorithm is also twisted to cover the possible need of the larger scale project which was defined in Electronic System Department at Linköping University. The extra twist added the harvesting mode to the algorithm too as harvesting was defined as part of the system in the larger project and every mote shall be able to recharge itself-through the energy source which is supposed to be provided for the network by another research work.

The whole algorithm is a simple concept that protect a mote from wasting its energy by regular environmental mistakes, for example. If a mote provides the “report of firing” in a remote part of a jungles, it shall not be triggered and goes to sending or receiving modes by a false alarm of thermal sensor which rises during a hot summer day.

This algorithm was implemented using Erlang language, showing different modes of a mote. We compared the result of this implementation with another implementation that only considers only two sending and receiving modes for a mote. The result shows that our algorithm has a higher bottleneck in computational part and suggests a higher complexity, though a mote implemented using this algorithm has a longer life time also. While a traditional mote, does not have such a computational bottleneck but suffers from shorter life time. This can show a weakness in our design too as our design suffers from a long response time and low sensitivity margin.

The whole design has put into test in a small network which was designed using SystemC. The result of this implementation shows that this scenario can be successfully used in Smart Dust networks.

8.1 Future work

Changing the mote behavior obviously is not the best and the most complicated solution for decreasing the power consumption in a Smart Dust network. Of course, there can be more works to be done to improve the algorithm which is suggested in this work but also looking to the network as a whole, can give more solutions to be worked on.

One of the most interesting feature that can make Smart Dust network adopted to, is AI or Artificial Intelligence. Earlier in this work, there is a short chapter on this subject however we couldn't work more detail in this area but it can offer really high measures for the applications that is adopted to these technologies. Based on experience and earlier studies, neural networks and fuzzy logic are two AI technologies that can be well applied to Smart Dust networks.

As it has mentioned in conclusion, this design suffers from a low triggering edge comparing with the traditional design. It also has a bottleneck on computational part of the design that can considerably increase the response time. This can be a major weakness for application with a really high margin of sensitivity and/or also applications which demand a fast reply. So implementing this algorithm using another programming language can be quite interesting.

Also this algorithm can be implemented using more than one programming languages and the results can be compared. This can show which language can serve Smart Dust network best in this case.

The other interesting area that can be well interesting is jamming which this document only has introduced the subject as an issue for smart Dust networks also.

8.2 Suggestions

As it has mentioned in conclusion, this design suffers from a low triggering edge comparing with the traditional design. It also has a bottleneck on computational part of the design that can considerably increase the response time. This can be a major weakness for application with a really high margin of sensitivity and/or also applications which demand a fast reply.

It is suggested to implement the same proposal with another programming language that can solves the issue.

This work does not have any specific focus on the communication mean which can well differentiate the results, so one really good and helpful area to put the focus on, under the same thesis title, can be the mean of communication that can be used by motes. Lots of research work that we have faced during this thesis work, focuses on optical communication specifically for the works that have been implemented in the real world applications. Though unfortunately this cannot be an option for most of the applications that are critical, for instance “firing report in a remote jungle” example which has been used many times as an example during this document.

9 REFERENCES

- [1] <http://en.wikipedia.org/wiki/Smartdust>
- [2] Jonathan McCarrel McCune, "Adaptability in sensor network".
- [3] Sung-Bae Cho, "Fusion of Fuzzy Logic with neural network".
- [4] Stephen Chiu, Sujeet Chand, "Self-organizing traffic control via Fuzzy Logic".
- [5] R. Narayanan, R. Udayakumar, K. Kumar, L. Subbaraj, "Quantification of congestion using Fuzzy Logic".
- [6] Zdravko Karakehayov, "Low-power design for smart dust network".
- [7] Yang Liu, "Wireless sensor network simulation".
- [8] Davide C. Black and Jack Donovan, "SystemC: from the ground up".
- [9] Joe Armstrong, "Programming Erlang, software for a concurrent world".
- [10] Erlang official website <http://www.Erlang.org/>.
- [11] Learn you some Erlang for great good! Link.
- [12] <http://en.wikipedia.org/wiki/Self-organization>
- [13] J.M. Kahn, R.H. Katz, K.S.J Pister, "Next century challenges: mobile network for 'smart dust'".
- [14] Tom Germano, <http://davis.wpi.edu/~matt/courses/soms/> hitted on 14th May 2011.
- [15] <http://www.tinyos.net/> hitted on 13th May 2011.
- [16] Ingo Schramm, <http://www.ingo-schramm.de/blog/archives/11-What-ErlangOTP-is-good-for-and-what-it-is-not.html>, hitted on May 16th 2011. Written in May 27th 2009.
- [17] Alex Pasternack, <http://www.motherboard.tv>, Mar 28, 2011, hitted on 23 May 2011.
- [18] Muneeb Ali, "A Brief History of Sensor Networks".
- [19] Udin Harun, <http://alkautsarpens.wordpress.com>, Hitted on 25th May.
- [20] Vasu Jolly and Shahram Latifi, "Comprehensive Study of Routing Management in Wireless Sensor Networks", Part1.
- [21] Chiara Buratti, Marco Martalo, Gianlugi Ferrari and Roberto Verdone, "Sensor Networks with IEEE 802.15.4 Systems Distributed Processing, MAC, and Connectivity".
- [22] <http://www.webopedia.com> hitted on 25th May 2011.
- [23] <http://www-bsac.eecs.berkeley.edu/programs/smartdust.html> Berkeley Sensor & Actuator Center, hitted on 26th May 2011.
- [24] http://www.computerhistory.org/internet_history/ Hitted on 27th May 2011.
- [25] Homayun Bakht, <http://www.computingunplugged.com/issues/issue200508/00001598001> Computing Unplugged Magazine, Hitted 27th May 2011.
- [26] Homayun Bakht, <http://www.computingunplugged.com/issues/issue200410/00001398001.html> Computing Unplugged Magazine, Hitted 27th May 2011.

- [27] Humayun Bakht, "History of Mobile Ad-hoc network".
- [28] Kay Romer, "Tracking Real-World Phenomena with Smart Dust".
- [29] Doug Steel, "Smart Dust – UH ISRC Technology Briefing".
- [30] Patrick Kinney, "ZigBee Technology: Wireless Control that Simply works".
- [31] S.S. Riaz Ahamed, "The role of Zigbee technology in future data communication system".
- [32] <http://www.nanotech-now.com/smartdust.htm>
- [33] http://edition.cnn.com/2010/TECH/05/03/smart.dust.sensors/index.html?_s=PM:TECH
- [34] <http://www.redicecreations.com/specialreports/smartdustmatrix.html>
- [35] Joseph M. Kahn, Randy Howard Katz, and Kristofer S. J. Pister, "Emerging Challenges: Mobile Networking for 'Smart Dust'", Journal of Communications and Networks, June 2000.
- [36] Mark Hempstead, Michael J. Lyons, David Brooks, and Gu-Yeon Wei, "Survey of Hardware Systems for Wireless Sensor Networks", Journal of Low Power Electronics, Vol.4, 1–10, 2008.
- [37] Ioannis Chatzigiannakis, Sotiris Nikolettseas and Paul Spirakis, "Efficient and Robust Protocols for Local Detection and Propagation in Smart Dust Networks", Springer Science + Business Media, Inc., Mobile Networks and Applications 10, 133–149, 2005.
- [38] <http://ftp.sunet.se/pub/lang/Erlang/download.html>
- [39] <http://www.thinkingparallel.com/2007/03/20/ten-questions-with-joe-armstrong-about-parallel-programming-and-Erlang/>
- [40] Yunbin Song, "Optical Communication Systems for Smart Dust", Master thesis report, Virginia Polytechnic Institute and State University, August 2002.
- [41] V. S. Hsu, J.M. Kahn, K. S. J. Pister, Wireless Communication for Smart Dust, 1998.
<http://robotics.eecs.berkeley.edu/~pister/SmartDust/>
- [42] J.M. Kahn, R.H. Katz, K. S. J. Pister, "Next Century Challenges: Mobile Networking for mart Dust".
<http://robotics.eecs.berkeley.edu/~pister/SmartDust/>
- [43] P. B. Chu, N. R. Lo, E. C. Berg, K. S. J. Pister, Optical Communication Using Micro CornerCube Reflectors, IEEE workshop on MEMS, 1997.
- [44] J.M. Kahn and J.R. Barry, "Wireless Infrared Communications", Proc. of the IEEE, pp. 265-298, February 1997 (Invited Paper).
- [45] K. S. J. Pister, J.M. Kahn, B. E. Boser, Wireless Networks of Millimeter-Scale Sensor Nodes, 1998,
<http://robotics.eecs.berkeley.edu/~pister/SmartDust/>
- [46] Kris Pister, Smart Dust: Autonomous sensing and communication in a cubic millimeter,
<http://robotics.eecs.berkeley.edu/~pister/SmartDust/>
- [47] B. Warneke, B. Atwood, K. S. J. Pister, Smart Dust Mote Forerunners, Proc. IEEE Inter. Conference on MEMS, 2001.
- [48] V. S. Hsu, J. M. Kahn, S. J. Pister, MEMS Corner Cube Retroreflector for Free-Space Optical Communications, 1991, <http://robotics.eecs.berkeley.edu/~pister/>
- [49] X. Zhu, V. S. Hsu, J.M. Kahn, Optical Modeling of MEMS Corner Cube Retroreflectors With Misalignment and Nonflatness, IEEE, 2002.
- [50] [12] P. B. Chu, N. R. Lo, E. C. Berg, K. S. J. Pister, Optical Communication Using Micro Corner Cube Reflectors, IEEE Inter. Workshop on MEMS, 1997.
- [51] "Self-Organizing Network (SON) - Introducing the Nokia Siemens Networks SON Suite – an efficient, future-proof platform for SON", Nokia Siemens Networks.

- [52] Magdalena Nohrborg, "Self-Organizing Networks", 3GPP, a global initiative
<http://www.3gpp.org/SON>
- [53] Ian Poole, "self-organizing Networks, SON", Radio-Electronics.com.
<http://www.radio-electronics.com/info/cellulartelecomms/self-organising-networks-son/basics-tutorial.php>
- [54] <http://javasim.codehaus.org/>

10 APPENDICES

10.1 Appendix 1: Erlang Codes

10.1.1 mote_bhv

%% This module describes the motes behaviour. You'll have 50 sec to decide
%% to which state you want to change the mote's mode in sleeping, sensing
%% and receiving mode.

```
-module(mote_bhv).  
-compile(export_all).
```

```
start() ->  
    spawn(fun() -> sleep() end).
```

```
high_sensed_signal_threshold(Mote) -> Mote ! high_sensed_signal_threshold.
```

```
more_sensed_signal(Mote) -> Mote ! more_sensed_signal.
```

```
no_more_sensed_signal(Mote) -> Mote ! no_more_sensed_signal.
```

```
high_received_signal_power(Mote) -> Mote ! high_received_signal_power.
```

```
more_received_signal(Mote) -> Mote ! more_received_signal.
```

```
no_more_received_signal(Mote) -> Mote ! no_more_received_signal.
```

```
sleep() ->  
    io:format("Mote is in sleeping mode, Energy usage of class III ~n"),  
    receive  
        high_sensed_signal_threshold ->  
            io:format("Mote is transferring to sensing mode ~n"),  
            sense();  
        high_received_signal_power ->  
            io:format("Mote is transferring to receiving mode ~n"),  
            rec()%%;  
        %% _ ->  
        %% sleep()  
        after 5000 ->  
            sleep()  
    end.
```

```
sense() ->  
    io:format("Mote is in sensing mode, Energy usage of class II ~n"),  
    receive
```

```

        more_sensed_signal ->
            io:format("Mote is transferring to sending mode ~n"),
            send();
        no_more_sensed_signal ->
            io:format("No more serious detection ~n"),
            sleep()
    after 50000 ->
        sleep()
end.

```

%% This function could not be named receive, as this word is a reserved word in Erlang.

```

rec() ->
    io:format("Mote is in receiving mode, Energy usage of class II ~n"),
    receive
        more_received_signal ->
            io:format("Mote is transferring to sending mode ~n"),
            send();
        no_more_received_signal ->
            io:format("No more signal received ~n"),
            sleep()
    after 50000 ->
        sleep()
end.

```

%% After sending the message, after 30 sec, mote will go back to its sleeping mode.

%% Here we suppose the sending time won't be more than 30sec.

```

send() ->
    io:format("Mote is in sending mode, Energy usage of class I ~n"),
    receive
        _ ->
            sleep()
    after 30000 ->
        sleep()
end.

```

%% Erlang shell commands for execution

%% cd("E:/LIU/Thesis/SmartDust Network/ErlangCodes/work/src/Ate1").

%% c(mote_bhv.erl).

%% Mote=mote_bhv:start().

%% mote_bhv:high_sensed_signal_threshold(Mote).

%% mote_bhv:more_sensed_signal(Mote).

10.1.2 base_station

%% This module is going to show the protocol that base station uses to show a proper
%% action for motes and users.

```
-module(base_station).
```

```
-compile(export_all).
```

```
-record(bsrec, {motes,          %% list of motes (mote_record)  
                clients}).
```

```
-record(mote, {id = "",  
              Pid,  
              remained_power,  
              mode = sleep}).
```

%% Initiation Functions...

```
start() ->  
    spawn(?MODULE, init, {}).
```

```
start_link() ->  
    spawn_link(?MODULE, init, {}).
```

%% In init, we define our mote to be of the type of ordered dictionary.

```
init() ->  
    loop(#Bsrec{motes=orddict:new(),  
                clients=orddict:new()}).
```

```
loop(B = #bsrec{}) ->  
    receive
```

%% Handling messages between base station and customer:

```
{Pid, MsgRef, {setup, Client}} ->  
    Ref = Erlang:monitor(process, Client),  
    %% The client information is now stored under the Ref name.  
    NewClients = orddict:store(Ref, Client, #Bsrec.clients),  
    Pid ! {MsgRef, ok},  
    loop(B#bsrec{clients=NewClients});
```

```
{Pid, MsgRef, {add, Id, Power, Mode}} ->  
    MotePid = mote_event:start_link(Id, Power),  
    NewMote = orddict:store(Id, #mote{id = Id,
```

Pid = MotePid,

```
remained_power=Power,
```

mode = Mode},
B#bsrec.motes),

```
Pid ! {MsgRef, ok},  
loop(B#bsrec{motes=NewMote});
```

```
{Pid, MsgRef, {delete, id}} ->  
    Motes = case orddict:find(Id, B#bsrec.motes) of  
        {ok, M} ->
```

```

        mote_event:delete(M#mote.Pid),
        orddict:erase(Id, B#bsrec.motes);
    error ->
        B#bsrec.motes
    end,
    Pid ! {MsgRef, ok},
    loop(B#bsrec{motes=Motes});

{Pid, MsgRef, {event_trigger, Id}} ->
    Motes = case orddict:find(Id, B#bsrec.motes) of
        {ok, M} ->
            mote_event:event_trigger(M#mote.Pid),
        error ->
            io:format("Wanted ID didn't found in the network,Please check your
mote ID again~p~n"),
            B#bsrec.motes
    end,
    Pid ! {MsgRef, ok},
    loop(B#bsrec{motes=Motes});

%% Handling messages between base satation and motes:

%% {done, Id} ->
%%     E = orddict:fetch(Id, B#bsrec.motes),

end.

```

10.1.3 mote_event

```

-module(mote_event).
%% -compile(export_all).
-export([start/2, start_link/2, delete/1]).
-export([init/3, loop/1]).

-record(mote, {server,
               id = "",
               remained_power=1, %% It has one joul per day.
               mode = sleep}).

%% Initialization
start (Id, Power) ->
    spawn(?MODULE, init, [self(),Id, Power, sleep]).

start_link (Id, Power) ->
    spawn_link(?MODULE, init, [self(),Id, Power, sleep]).

init(Server, Id, Power) ->
    loop(#mote{server = Server,
               id = Id,
               remained_power = Power_calc(power)-0.02, %% This part should be replaced
with a proper power_calc function.
               mode = sleep}).

```



```
%% Internal functions
```

```
delete(Id) ->  
    Ref=Erlang:monitor(process, Id),  
    Id ! {self(), Ref, delete},  
    receive  
        {Ref, ok} ->  
            Erlang:demonitor(Ref, [flush]),  
            ok;  
        {'DOWN', Ref, process, Id, _Reason} ->  
            ok  
    end.
```

```
event_trigger(Id) ->  
    receive  
        {} ->
```

```
%% Loop and it required functions  
loop(#mote{server = Server}) ->  
    receive  
        {Server, Ref, delete} ->  
            Server ! {Ref, ok}  
    end.
```

```
%% Internal functions  
%% power_calc(Power) ->
```

10.1.4 base_station_protocol

```
-module(base_station).  
-compile(export_all).
```

```
-----  
%% Initiation Functions...
```

```
start()->
```

```
....
```

```
start_link() ->
```

```
....
```

```
init() ->
```

```
....
```

```
-----  
%% Internal functions that should be handled by base station, needed and called in loop function.
```

```
terminate() ->
```

```
....
```

```
setup(...) ->
```

```
....
```

```
add_mote(...) ->
```

```
....
```

```
delete(...) ->
```

```
....
```

```
event_trigger(...) ->
```

```
....
```

```
-----  
%% Loop function for handling the messages by base station...
```

```
loop(bsrec) ->
```

```
    receive
```

```
%% Handling messages between base station and customer:
```

```
        {Pid, MsgRef, {setup, Client}} ->
```

```
        ...
```

```
        {Pid, MsgRef, {add, Id, Power, Mode}} ->
```

```
        ...
```

```
        {Pid, MsgRef, {delete, Id}} ->
```

```
        ...
```

```
        {Pid, MsgRef, {event_trigger, Id}} ->
```

```
        ...
```

```
%% Handling messages between base station and motes:
```

```
{send, Id} ->
```

```
...
```

```
%% Handling messages between motes:
```

```
    {senese, Id, Power, Mode} ->
```

```
        ...
```

```
    {receive, Id, Child_Id} ->
```

```
        ...
```

```
    {send, Id, Parent_Id} ->
```

```
        ...
```

```
%% Others...
```

```
    shutdown ->
```

```
        ...
```

```
    {'DOWN', Ref, process, _Pid, _Reason} ->
```

```
        ...
```

```
    code_change ->
```

```
        ...
```

```
    Unknown ->
```

```
        io:format("Unknown message: ~p~n",[Unknown]),
```

```
        loop(bsrec)
```

```
end.
```

```
%% Internal Functions
```

10.1.5 user_protocol

```
-module(user_protocol).
```

```
-compile(export_all).
```

10.2 Appendix 2: C-Codes

10.2.1 Mote.cc

```
#include <iostream>
#include <mote.h>
using std::cout;
using std::endl;

//Mote::Mote(sc_module_name name, int x, int y,int neighbour_mote,int mote_number,int receive_mote,int
output_mote,int sensor_input)
Mote::Mote(sc_module_name name)
    : sc_module(name)

{

counter=0;energy=10;
SC_THREAD(mote_sleep);
SC_METHOD(mote_send);
dont_initialize();
SC_METHOD(mote_harvest);
dont_initialize();
SC_METHOD(mote_sense);
dont_initialize();
sensitive << higher_threshold;
SC_METHOD(mote_receive);
dont_initialize();
sensitive <<received_signal;
mote_array[1]=0;
mote_array[2]=1;
mote_array[3]=2;
mote_array[4]=3;
mote_array[5]=6;
mote_array[6]=7;
mote_array[7]=1;
mote_array[8]=6;
mote_array[9]=8;
}

void Mote::mote_harvest()
{

while(energy < 10)
{
energy =energy + 0.5;
}
cout<< sc_time_stamp() <<" " << name()<<" with coordinates "<< x <<" and "<<y<< " is out of harvesting
mode"
        << endl;
}
```

```

void Mote::mote_sense()
{
cout<< sc_time_stamp() << " " <<name()<<" with coordinates "<< x <<" and "<<y<< " is in sensing mode-
counter ="<<counter<<" and the Available Energy = "<<energy
    << endl;

counter++;
if ( counter >=3)
{
counter=0;
mote_send();
}
energy=energy-0.5;

}

```

```

void Mote::mote_send()
{
int i=9;
int neighbour_mote_temp;
//wait(2,SC_MS);
energy=energy-1.5;
neighbour_mote_temp=neighbour_mote;
output_mote=neighbour_mote;
if ( neighbour_mote == 0 ){
cout<< sc_time_stamp() <<" " << name() << ": Message Sent to the BaseStation"<<" and the Available
Energy = "<<energy
<< endl;
cout <<"-----" <<endl;
}
else if(harvest_parent_flag==true)
{
cout<< sc_time_stamp() <<" " << name() << ":Parent mote "<<neighbour_mote<<" is into harvesting-
Rerouting to the next possible mote "<<mote_array[neighbour_mote]<<" and the Available Energy =
"<<energy<< endl;

if( mote_array[neighbour_mote]== 0 or lineage !=0)
{
cout<< sc_time_stamp() <<" " << name() << ": Sending message to the BaseStation -
lineage"<<lineage<<" and the Available Energy = "<<energy<<endl;
cout <<"-----" <<endl;
}
lineage.write(mote_array[neighbour_mote]);
}

//cout<< sc_time_stamp() << name() << ": Mote is in sending
mode"<<output_mote<<"hjj"<<neighbour_mote
    // << endl;

}

```

```

void Mote::mote_sleep()
{
int receive_mote_prev;
int lineage_prev;

```

```

bool sensor_input_prev;
int count=0;

for(;;){
if(energy <=2){
cout<< sc_time_stamp() <<" " << name()<<" with coordinates "<< x <<" and "<<y<<" is in harvesting
mode, WARNING: This mote is no longer available in the network!"

                << endl;
harvest_flag.write(true);
wait(900,SC_MS);
mote_harvest();
harvest_flag.write(false);
}

if( ((receive_mote==mote_number or receive_mote== 2*mote_number) and (receive_mote !
=receive_mote_prev) ) or ( (lineage==mote_number ) and (lineage !=lineage_prev)))
{
//cout << sc_time_stamp() <<"inside sleeping Receive" << name()<<receive_mote<<"//receive//"<<
receive_mote_prev << "//output/"<< output_mote <<endl;
received_signal.notify();

}
//cout << sc_time_stamp() <<"inside sleeping fine" << name()<<sensor_input<<"//"/"<< sensor_input_prev
<< endl;
if((sensor_input==true) and(sensor_input!=sensor_input_prev ) )
{

higher_threshold.notify();
}

//energy=energy-0.001;
receive_mote_prev=receive_mote;
lineage_prev=lineage;
sensor_input_prev=sensor_input;
output_mote=0;
wait(1,SC_MS);

        }

}

void Mote::mote_receive()
{

//for(;;){
//wait(received_signal);
cout<< sc_time_stamp() <<" " <<name()<<" with coordinates "<< x <<" and "<<y<<" is in receiving
mode"<<" and the Available Energy = "<<energy
                << endl;

energy=energy-1;
mote_send();
//wait(3,SC_MS);
//        }
}

```

```

mote_gen.cc
#include "Mote_gen.h"
#include <iostream>
#include <cstdlib>
#include <ctime>
using std::cout;
using std::endl;
Mote_gen::Mote_gen(sc_module_name name)
: sc_module(name)
{
    SC_THREAD(gen);
    SC_METHOD(output);
    dont_initialize();
    //sensitive << traffic_event;
}

void Mote_gen::output(int count,int flag)
{
    bool binary ='1';
    if( flag != 1) {
        //cout <<"-----" <<endl;

        switch (count) {
        case 0 :
            sensor_input_1.write(false);
            sensor_input_2.write(false);
            sensor_input_3.write(false);
            sensor_input_4.write(false);
            sensor_input_5.write(false);
            sensor_input_6.write(false);
            sensor_input_7.write(false);
            sensor_input_8.write(false);
            sensor_input_9.write(false);
            //wait(1,SC_NS);
            //cout << sc_time_stamp() <<" inside gen Zero Initialisation---" <<count<< endl;
            break;
        case 1 :
            sensor_input_1.write(true);
            //cout << sc_time_stamp() <<" inside gen 1 Initialisation---" <<count<< endl;
            //wait(1,SC_NS);
            break;
        case 2 :
            sensor_input_2.write(true);
            break;
        case 3 :
            sensor_input_3.write(true);
            break;
        case 4 :
            sensor_input_4.write(true);
            break;
        case 5 :
            sensor_input_5.write(true);
            break;
        case 6 :
            sensor_input_6.write(true);

```

```

break;
case 7 :
sensor_input_7.write(true);
break;
case 8 :
sensor_input_8.write(true);
break;
case 9 :
sensor_input_9.write(true);
}
}
}
void Mote_gen::gen()
{
//cout << "inside gen" << endl;
// Counter of 16 used to test all possible combinations
static int count=1;
int random=0;
// Random number chosen between 0 and 16.
srand(time(0));
random=rand()%9;
for(;;){

cout <<sc_time_stamp() << " Triggering Sensor for Mote ==> " <<count<< endl;
output(0,0);
    wait(10, SC_MS);
output(count,0);
    wait(100, SC_MS);
output(0,0);
wait(10, SC_MS);
output(count,0);
    wait(100, SC_MS);
output(0,0);
wait(10, SC_MS);
output(count,0);
    wait(100, SC_MS);
//count=9;
//traffic_event.notify();
    if ( count == 9 ) {
cout << "Normal scenarios tested for all Motes - WoooooHoooooooo" << endl;
    cout <<"-----" <<endl;
    count=0;
output(0,0);
wait(1, SC_MS);
cout << "starting to send RANDOM Mote with combination value in Integer :="
<<random<< endl;
output(random,0);
wait(100, SC_MS);
output(0,0);
wait(10, SC_MS);
output(random,0);
    wait(100, SC_MS);
output(0,0);
wait(10, SC_MS);
output(random,0);
wait(50, SC_MS);

```



```
        cout <<"-----END OF PROGRAM-----" <<endl;
exit(0);
}
    count = count + 1;
}
}
```

```

mote_main.cc
#include <systemc.h>
#include "mote.h"
#include "Mote_gen.h"
#include <iostream>
#include <cstdlib>
#include <ctime>
// #include <cassert> // for assert(bool)
int sc_main(int argc, char **argv)
{ // Two arguments: (1) the initial counter value, and
  // (2) the simulation time in seconds
  // assert(argc == 3);
  // int init_value = atoi(argv[1]);
  // sc_time sim_time( atof(argv[2]), SC_SEC);
  sc_time sim_time(12, SC_SEC);
  // instantiate module(s)
  // channels
  sc_signal<bool> sensor_input_1;
  sc_signal<bool> sensor_input_2;
  sc_signal<bool> sensor_input_3;
  sc_signal<bool> sensor_input_4;
  sc_signal<bool> sensor_input_5;
  sc_signal<bool> sensor_input_6;
  sc_signal<bool> sensor_input_7;
  sc_signal<bool> sensor_input_8;
  sc_signal<bool> sensor_input_9;
  sc_signal<int> output_mote_1;
  sc_signal<int> output_mote_2;
  sc_signal<int> output_mote_3;
  sc_signal<int> output_mote_4;
  sc_signal<int> output_mote_5;
  sc_signal<int> output_mote_6;
  sc_signal<int> output_mote_7;
  sc_signal<int> output_mote_8;
  sc_signal<int> output_mote_9;
  sc_signal<int> output_mote_10;
  sc_signal<int> output_mote_11;
  sc_signal<int> x1;
  sc_signal<int> y1;
  sc_signal<int> x2;
  sc_signal<int> y2;
  sc_signal<int> x3;
  sc_signal<int> y3;
  sc_signal<int> x4;
  sc_signal<int> y4;
  sc_signal<int> x5;
  sc_signal<int> y5;
  sc_signal<int> x6;
  sc_signal<int> y6;
  sc_signal<int> x7;
  sc_signal<int> y7;
  sc_signal<int> x8;
  sc_signal<int> y8;
  sc_signal<int> x9;
  sc_signal<int> y9;
  sc_signal<int> neighbour_mote1;

```

```

sc_signal<int> neighbour_mote2;
sc_signal<int> neighbour_mote3;
sc_signal<int> neighbour_mote4;
sc_signal<int> neighbour_mote5;
sc_signal<int> neighbour_mote6;
sc_signal<int> neighbour_mote7;
sc_signal<int> neighbour_mote8;
sc_signal<int> neighbour_mote9;
sc_signal<int>
harvest_mote0,harvest_mote1,harvest_mote2,harvest_mote3,harvest_mote4,harvest_mote5,harvest_mote
6,harvest_mote7,harvest_mote8,harvest_mote9;
sc_signal<int> lineage;
sc_signal<int> mote_number1;
sc_signal<int> mote_number2;
sc_signal<int> mote_number3;
sc_signal<int> mote_number4;
sc_signal<int> mote_number5;
sc_signal<int> mote_number6;
sc_signal<int> mote_number7;
sc_signal<int> mote_number8;
sc_signal<int> mote_number9;
sc_signal<int> mote_number10;
Mote_gen gen("Mote_gen");

```

```

gen(sensor_input_1,sensor_input_2,sensor_input_3,sensor_input_4,sensor_input_5,sensor_input_6,senso
r_input_7,sensor_input_8,sensor_input_9);

```

```

x1=1;
y1=0;
neighbour_mote1=0;
mote_number1=1;
//output_mote_11= output_mote_2 | output_mote_7;
Mote Mote1("Mote1");
Mote1(x1,y1,neighbour_mote1,mote_number1,output_mote_2,output_mote_1,sensor_input_1,harvest_mot
e1,harvest_mote0,lineage);
x2=1;
y2=2;
neighbour_mote2=1;
mote_number2=2;
Mote Mote2("Mote2");
Mote2(x2,y2,neighbour_mote2,mote_number2,output_mote_3,output_mote_2,sensor_input_2,harvest_mot
e2,harvest_mote1,lineage);
x3=1;
y3=4;
neighbour_mote3=2;
mote_number3=3;
Mote Mote3("Mote3");
Mote3(x3,y3,neighbour_mote3,mote_number3,output_mote_4,output_mote_3,sensor_input_3,harvest_mot
e3,harvest_mote2,lineage);
x4=3;
y4=5;
neighbour_mote4=3;
mote_number4=4;
Mote Mote4("Mote4");

```

```

Mote4(x4,y4,neighbour_mote4,mote_number4,output_mote_5,output_mote_4,sensor_input_4,harvest_mote4,harvest_mote3,lineage);
x5=5;
y5=5;
neighbour_mote5=6;
mote_number5=5;
Mote Mote5("Mote5");
Mote5(x5,y5,neighbour_mote5,mote_number5,output_mote_6,output_mote_5,sensor_input_5,harvest_mote5,harvest_mote6,lineage);
x6=4;
y6=3;
neighbour_mote6=7;
mote_number6=6;
Mote Mote6("Mote6");
//output_mote_10=output_mote_5 |output_mote_8;

Mote6(x6,y6,neighbour_mote6,mote_number6,output_mote_5,output_mote_6,sensor_input_6,harvest_mote6,harvest_mote7,lineage);
x7=3;
y7=1;
neighbour_mote7=1;
mote_number7=7;
Mote Mote7("Mote7");
Mote7(x7,y7,neighbour_mote7,mote_number7,output_mote_6,output_mote_2,sensor_input_7,harvest_mote7,harvest_mote1,lineage);
x8=5;
y8=3;
neighbour_mote8=6;
mote_number8=8;
Mote Mote8("Mote8");
Mote8(x8,y8,neighbour_mote8,mote_number8,output_mote_9,output_mote_5,sensor_input_8,harvest_mote8,harvest_mote6,lineage);
x9=4;
y9=6;
neighbour_mote9=8;
mote_number9=9;
Mote Mote9("Mote9");
Mote9(x9,y9,neighbour_mote9,mote_number9,output_mote_10,output_mote_9,sensor_input_9,harvest_mote9,harvest_mote8,lineage);

sc_start(sim_time); // start simulation
return 0;
}

```

10.2.2 mote.h

```
#ifndef MOTE_H
#define MOTE_H
#include <systemc.h>

SC_MODULE(Mote) {
    int counter;
    double energy;
    int mote_array[10];
    SC_HAS_PROCESS(Mote);
    sc_in<int> x;
    sc_in<int> y;
    sc_in<int> neighbour_mote;
    sc_in<int> mote_number;
    sc_in<int> receive_mote;
    sc_out<int> output_mote;
    sc_in<bool> sensor_input;
    sc_out<int> harvest_flag;
    sc_in<int> harvest_parent_flag;
    sc_inout<int> lineage;
    Mote(sc_module_name name);
    void mote_sleep();
    void mote_sense();
    void mote_send();
    void mote_receive();
    void mote_harvest();
    sc_event higher_threshold;
    sc_event received_signal;

};
#endif
```

10.2.3 mote_gen.h

```
#ifndef MOTE_GEN_H
#define MOTE_GEN_H
#include <systemc.h>
SC_MODULE(Mote_gen) { SC_HAS_PROCESS(Mote_gen);
sc_out<bool> sensor_input_1;
sc_out<bool> sensor_input_2;
sc_out<bool> sensor_input_3;
sc_out<bool> sensor_input_4;
sc_out<bool> sensor_input_5;
sc_out<bool> sensor_input_6;
sc_out<bool> sensor_input_7;
sc_out<bool> sensor_input_8;
sc_out<bool> sensor_input_9;
//sc_event traffic_event ;
Mote_gen(sc_module_name name);
void gen();
void output(int count,int flag);
};
#endif // TRAFFIC_H
```

10.2.4 out.txt

0 s Triggering Sensor for Mote ==> 1

11 ms Mote1 with coordinates 1 and 0 is in sensing mode-counter =0 and the Available Energy = 10

121 ms Mote1 with coordinates 1 and 0 is in sensing mode-counter =1 and the Available Energy = 9.5

231 ms Mote1 with coordinates 1 and 0 is in sensing mode-counter =2 and the Available Energy = 9

231 ms Mote1: Message Sent to the BaseStation and the Available Energy = 7.5

330 ms Triggering Sensor for Mote ==> 2

341 ms Mote2 with coordinates 1 and 2 is in sensing mode-counter =0 and the Available Energy = 10

451 ms Mote2 with coordinates 1 and 2 is in sensing mode-counter =1 and the Available Energy = 9.5

561 ms Mote2 with coordinates 1 and 2 is in sensing mode-counter =2 and the Available Energy = 9

562 ms Mote1 with coordinates 1 and 0 is in receiving mode and the Available Energy = 7

562 ms Mote1: Message Sent to the BaseStation and the Available Energy = 4.5

660 ms Triggering Sensor for Mote ==> 3

671 ms Mote3 with coordinates 1 and 4 is in sensing mode-counter =0 and the Available Energy = 10

781 ms Mote3 with coordinates 1 and 4 is in sensing mode-counter =1 and the Available Energy = 9.5

891 ms Mote3 with coordinates 1 and 4 is in sensing mode-counter =2 and the Available Energy = 9

892 ms Mote2 with coordinates 1 and 2 is in receiving mode and the Available Energy = 7

893 ms Mote1 with coordinates 1 and 0 is in receiving mode and the Available Energy = 4.5

893 ms Mote1: Message Sent to the BaseStation and the Available Energy = 2

894 ms Mote1 with coordinates 1 and 0 is in harvesting mode, WARNING: This mote is no longer available in the network!

990 ms Triggering Sensor for Mote ==> 4

1001 ms Mote4 with coordinates 3 and 5 is in sensing mode-counter =0 and the Available Energy = 10

1111 ms Mote4 with coordinates 3 and 5 is in sensing mode-counter =1 and the Available Energy = 9.5

1221 ms Mote4 with coordinates 3 and 5 is in sensing mode-counter =2 and the Available Energy = 9

1222 ms Mote3 with coordinates 1 and 4 is in receiving mode and the Available Energy = 7

1223 ms Mote2 with coordinates 1 and 2 is in receiving mode and the Available Energy = 4.5

1223 ms Mote2:Parent mote 1 is into harvesting-Rerouting to the next possible mote 0 and the Available Energy = 2

1223 ms Mote2: Sending message to the BaseStation - lineage0 and the Available Energy = 2

1224 ms Mote2 with coordinates 1 and 2 is in harvesting mode, WARNING: This mote is no longer available in the network!

1320 ms Triggering Sensor for Mote ==> 5

1331 ms Mote5 with coordinates 5 and 5 is in sensing mode-counter =0 and the Available Energy = 10

1441 ms Mote5 with coordinates 5 and 5 is in sensing mode-counter =1 and the Available Energy = 9.5

1551 ms Mote5 with coordinates 5 and 5 is in sensing mode-counter =2 and the Available Energy = 9

1552 ms Mote6 with coordinates 4 and 3 is in receiving mode and the Available Energy = 10

1553 ms Mote7 with coordinates 3 and 1 is in receiving mode and the Available Energy = 10

1553 ms Mote7:Parent mote 1 is into harvesting-Rerouting to the next possible mote 0 and the Available Energy = 7.5

1553 ms Mote7: Sending message to the BaseStation - lineage0 and the Available Energy = 7.5

1650 ms Triggering Sensor for Mote ==> 6

1661 ms Mote6 with coordinates 4 and 3 is in sensing mode-counter =0 and the Available Energy = 7.5

1771 ms Mote6 with coordinates 4 and 3 is in sensing mode-counter =1 and the Available Energy = 7

1794 ms Mote1 with coordinates 1 and 0 is out of harvesting mode

1881 ms Mote6 with coordinates 4 and 3 is in sensing mode-counter =2 and the Available Energy = 6.5

1882 ms Mote7 with coordinates 3 and 1 is in receiving mode and the Available Energy = 7.5

1883 ms Mote1 with coordinates 1 and 0 is in receiving mode and the Available Energy = 10

1883 ms Mote1: Message Sent to the BaseStation and the Available Energy = 7.5

1980 ms Triggering Sensor for Mote ==>> 7
1991 ms Mote7 with coordinates 3 and 1 is in sensing mode-counter =0 and the Available Energy = 5
2101 ms Mote7 with coordinates 3 and 1 is in sensing mode-counter =1 and the Available Energy = 4.5
2124 ms Mote2 with coordinates 1 and 2 is out of harvesting mode
2211 ms Mote7 with coordinates 3 and 1 is in sensing mode-counter =2 and the Available Energy = 4
2212 ms Mote7 with coordinates 3 and 1 is in harvesting mode, WARNING: This mote is no longer available in the network!
2212 ms Mote1 with coordinates 1 and 0 is in receiving mode and the Available Energy = 7.5
2212 ms Mote1: Message Sent to the BaseStation and the Available Energy = 5

2310 ms Triggering Sensor for Mote ==>> 8
2321 ms Mote8 with coordinates 5 and 3 is in sensing mode-counter =0 and the Available Energy = 10
2431 ms Mote8 with coordinates 5 and 3 is in sensing mode-counter =1 and the Available Energy = 9.5
2541 ms Mote8 with coordinates 5 and 3 is in sensing mode-counter =2 and the Available Energy = 9
2542 ms Mote6 with coordinates 4 and 3 is in receiving mode and the Available Energy = 4.5
2542 ms Mote6:Parent mote 7 is into harvesting-Rerouting to the next possible mote 1 and the Available Energy = 2
2543 ms Mote6 with coordinates 4 and 3 is in harvesting mode, WARNING: This mote is no longer available in the network!
2543 ms Mote1 with coordinates 1 and 0 is in receiving mode and the Available Energy = 5
2543 ms Mote1: Message Sent to the BaseStation and the Available Energy = 2.5

2640 ms Triggering Sensor for Mote ==>> 9
2651 ms Mote9 with coordinates 4 and 6 is in sensing mode-counter =0 and the Available Energy = 10
2761 ms Mote9 with coordinates 4 and 6 is in sensing mode-counter =1 and the Available Energy = 9.5
2871 ms Mote9 with coordinates 4 and 6 is in sensing mode-counter =2 and the Available Energy = 9
2872 ms Mote8 with coordinates 5 and 3 is in receiving mode and the Available Energy = 7
2872 ms Mote8:Parent mote 6 is into harvesting-Rerouting to the next possible mote 7 and the Available Energy = 4.5
2872 ms Mote8: Sending message to the BaseStation - lineage1 and the Available Energy = 4.5

Normal scenarios tested for all Motes - WoooooHoooooooooooo

starting to send RANDOM Mote with combination value in Integer :=2
2972 ms Mote2 with coordinates 1 and 2 is in sensing mode-counter =0 and the Available Energy = 10
3082 ms Mote2 with coordinates 1 and 2 is in sensing mode-counter =1 and the Available Energy = 9.5
3112 ms Mote7 with coordinates 3 and 1 is out of harvesting mode
3112 ms Mote7 with coordinates 3 and 1 is in receiving mode and the Available Energy = 10
3113 ms Mote1 with coordinates 1 and 0 is in receiving mode and the Available Energy = 2.5
3113 ms Mote1: Message Sent to the BaseStation and the Available Energy = 0

3114 ms Mote1 with coordinates 1 and 0 is in harvesting mode, WARNING: This mote is no longer available in the network!
3192 ms Mote2 with coordinates 1 and 2 is in sensing mode-counter =2 and the Available Energy = 9
3192 ms Mote2:Parent mote 1 is into harvesting-Rerouting to the next possible mote 0 and the Available Energy = 7.5
3192 ms Mote2: Sending message to the BaseStation - lineage7 and the Available Energy = 7.5

-----END OF PROGRAM-----