BIRZEIT UNIVERSITY

FACULTY OF INFORMATION TECHNOLOGY

Actor Node Positioning in Wireless Sensor Networks

تحديد مواقع العقد المتحكمة الفاعلة في شبكة الاستشعار اللاسلكية

This Thesis was submitted in partial fulfillment of the requirements for the Master's Degree in Scientific Computing from the Faculty of Graduate Studies at Birzeit University, Palestine

By:

Nemer Shaikh

Advisor:

Dr. Mohammad Jubran

Birzeit University

30 April 2013, Ramallah, Palestine

Actor Node Positioning in Wireless Sensor Networks

تحديد مواقع العقد المتحكمة الفاعلة في شبكة الاستشعار اللاسلكية

By Nemer Shaikh

This thesis was defended by the successfully student on 30 April 2013 and approved by:

Committee Member	Signature
Dr. Mohammad Jubran	---
Dr. Iyad Tomar	---
Dr. Abualseoud Hanani	---

Acknowledgements

Thanks are due to each and every person who helped out to make this work becomes a fact; I really appreciate the support and patience of Dr. Mohammad Jubran at Birzeit University, who supervised this work.

Thanks to my parents and family who supported me to the end, especially during the last days of this work.

Nemer Shaikh

Ramallah, Palestine

30 of April 2013

Table of Contents

List of Figures

List of Tables

Abstract

Wireless sensor networks drawn huge attention in the recent years due to their vast implementation scenarios and possible applications in many fields in our daily lives, opening the venue for a new era of technological evolution in human history.

Still WSN suffer from many constrains affecting their performance and operation, such as the energy consumption due to the fact that they are battery powered, latency, coverage and connectivity of the network, which may limits the network life time and operational efficiency.

Careful placement of WSN nodes has been proved to solve some of these issues, a new area in this topic, is actor nodes placement problem, our work falls into this category.

This work focuses on decreasing the average energy consumption of the WSN nodes, and the average communication delay in the WSN network by clustering the network and optimizing the actor's location within each cluster to achieve a better and more efficient network topology.

The Delay and Energy saving by Actuator positioning algorithM (DEMA) outperforms the average delay and average energy consumption when benchmarked with random, uniform, and COLA algorithms with around 25% reduction in the average communication delay, and around 40% in the average energy consumption. It also has a topology maintenance phase to readjust the network to optimal time periodically or when a percent of the node fail. Also it's built on the fact that the nodes use necessary amount of energy to communicate and not full energy mode as in other algorithms.

الملخص

شبكات المجسات اللاسلكية تعد فرعاً جديداً في مجال الاتصالات يؤسس لثورة قادمة في مجالات عدة في العلوم والتكنولوجيا والطب، حيث أنها تفتح الباب لدراسة العديد من الظواهر الطبيعية والعمليات الصناعية بالاضافة لبعض الامراض بطرق جديدة وعن قرب بتوفير قراءات وقياست لم يكن من السهل الحصول عليها في الماضي.

لكن هذا النوع من الشبكات اللاسلكية لا يزال يعاني من مشاكل كثيرة تحد من ادائها وامكانية استخدامها في الكثير من التطبيقات، نتيجة لعدد من العوامل وأهمها أن عناصر هذه الشبكة تعتمد على البطاريات كمصدر للطاقة وبالتالي فان عمل الشبكة محكوم بعمر البطارية وسرعة استهلاك محتواها من الطاقة، بالاضاف ة الى مجموعة من المحددات الاخرى مثل تحسين تغطية الشبكة في المنطقة المستهدفة، وجعل الشبكة مترابطة بشكل يتيح مجال للمناورة في حال انقطاع بعض الروابط في الشبكة.

إن موضعة العناصر في هذا النوع من الشبكات أثبت أنه بإمكانه حل العديد من المشاكل السالفة الذكر مما يساهم في إطالة عمر الشبكة وتحسين عملها، في الاونة الأخيرة ظهر توجه جديد نحو موضعة العناصر الفاعلة في مواقع تحسن من اداء الشبكة ، ويمكن تصنيف عملنا في هذه الرسالة ضمن هذه الفئة.

تم التركيز في هذا العمل على موضعة العناصر الفاعلة في الشبكة في اماكن تحقق شرطين اساسيين، اولهما تحقيق اقل معدل لعدد النقلات اللازمة للاتصال بين اي عنصر في الشبكة واقرب عنصر فعال مما يقلل المدة اللازمة لوصول البيانات داخل الشبكة، وثانيهما تحقيق اقل معدل لاستهلاك الطاقة اثناء عملية الاتصال، مما ينعكس ايجاباً عل اداء الشبكة و يطيل عمرها التشغيلي.

الخوازمية التي تم تطويرها أثناء مسار هذه الرسالة حققت نتائج مميزة في مجالي توفير الطاقة و تسريع الاتصال بين عناصر الشبكة و العناصر الفاعلة فيها، حيث تمت عملية مقارنة اداء الخورزمية مع خوارزميات اخرى تمثلت بالتوزيع العشوائي و التوزيع المنتظم للعناصر الفاعلة في الشبكة، كذلك تمت المقارنة مع خوارزمية COLA المتخصصة في هذا المجال حيث بدى تفوق الخوارزمية المطورة في هذا البحث واضحاً حيث تراوحت نسبة التحسين من 25% الى 50%.

كما تم اضافة وظيفتين جديدتين لا توجدان في خوارزميات اخرى في هذا المجال، اولهما استخدام نموذج لاستهلاك الطاقة يتناسب طردياً مع مسافة الاتصال بين اي عنصرين في الشبكة وليس العمل بطاقة كاملة اثناء الاتصال، وثانيهما اضافة مرحلة اعادة هيكلة الشبكة عندما تفقد نسبة من عناصرها مما يحافظ على بقاء الشبكة في الوضعية المثلى اثناء عملها.

1. Introduction

This chapter provides an introduction to the WSN node placement topic and sets the framework of the thesis.

- ☐ Introduction to WSN
- ☐ WSN node placement problem
- ☐ Thesis scope and problem statement

Wireless sensor networks are an emerging technology with huge implementation potential in many civilian and military applications such as, battle field surveillance, habitat monitoring, space exploration and many other applications like telemedicine and others, as will be shown in later sections.

Wireless sensor networks are very constrained in terms of power consumption. Many algorithms and techniques have been developed in an effort to increase the network lifetime by making all the network structures and functionalities very efficient in utilizing available energy resources.

Wireless sensor networks received a lot of research interest during the last ten years due to the diverse possible applications of such networks.

1.1 Brief History of WSN

In the beginning of the 1990's, there was a huge development in three main sectors:

- Communication systems were evolving rapidly.
- Computer hardware speeds were alot ahead of Moor's law.
- MEMS technology was producing new and much smarter sensors.

And so came the idea of integrating those three technologies in a single product, which will in turn facilitate many applications, which later in 1998 was formulated in the birth of wireless sensor networks concept.

In 2003 the first trial to establish a standard for WSN was done by the IEEE. The standard name was IEEE 802.15.4. Later in 2004 Zigbee alliance was born, which included at the beginning 22 companies, today more than 150 companies are part of the alliance, all of them share the same interest in developing WSN technology and integrating it into their products and solutions.

Later on 2006 the IEEE published the latest standard for WSN under the name IEEE 802.15.4-2006 and in 2007 the Zigbee alliance published the latest standard Zigbee PRO which are under continuous development until today.

1.2 Technical Overview

The network is a collection of huge number of sensors for different purposes that communicate between each other, according to orders coming from the command center through the nearest base station as shown in Figure 1.1 below.

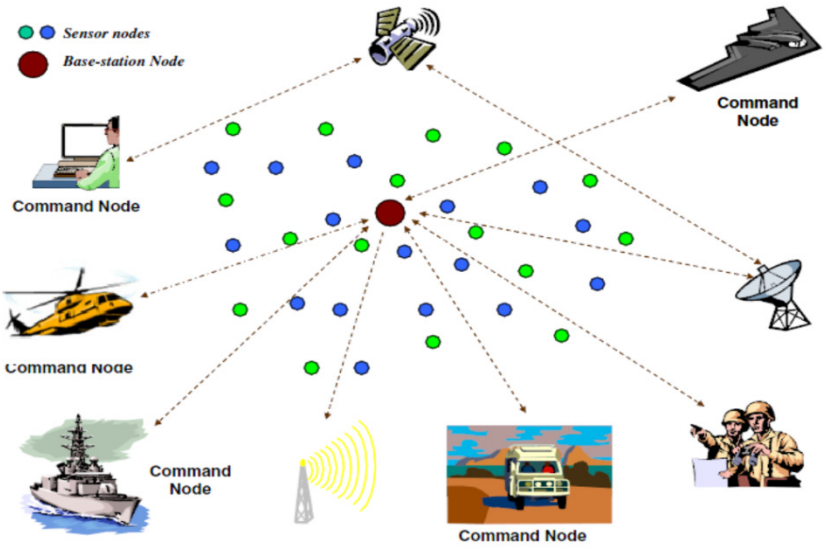


Figure 1.1: General WSN structure [3]

Different types of sensor nodes have been developed to overcome available network constrains. Section 1.2.1 discusses the sensor nodes components and node types

1.2.1 Sensor Nodes

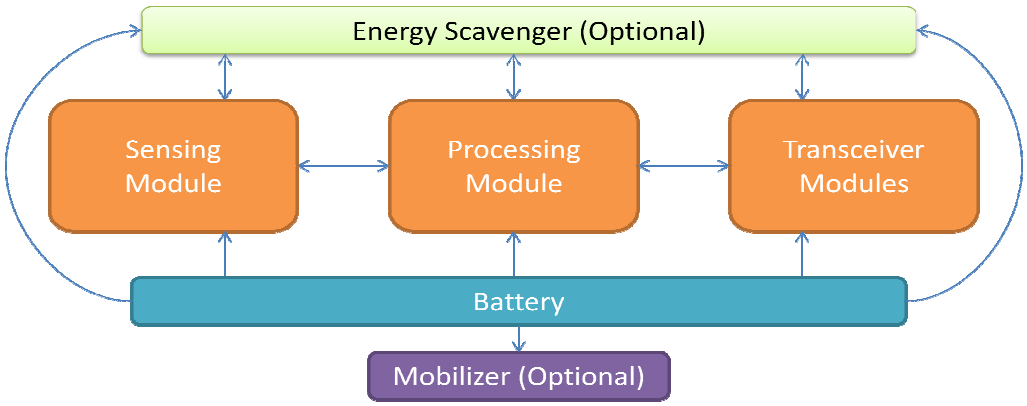


Figure 1.2: Sensor node architecture [9]

1.2.1.1 Used Sensors on the Node

A sensor node is equipped with different types of sensors to perceive and measure the phenomena and changes taking place in the surrounding deployment environment, such as temperature and acoustic sensors or even tiny cameras.

1.2.1.2 Processing and Storage

After the sensing unit captures the required parameter, the raw measurements are stored and forwarded to the processing unit of the node, where the size of data to be transmitted is reduced to the minimum required using different data manipulation methods according to the application. The processing unit supervises and controls the node operation.

1.2.1.3 Transceiver

Another main component is the communication hardware and software. Some implementations of WSN used optical and infrared transceivers which suffered from a big problem since they require line of sight (LOS) between different nodes to be able to cooperate. On the other hand Bluetooth and radio frequencies based transmission was successfully deployed in the majority of implementation since its cheaper and more robust, not to mention that it doesn't need LOS between the nodes.

1.2.1.4 Energy Unit

As always an energy unit is a must to power up the whole node. Normally in WSN it is battery supported with the possibility of adding energy scavenging hardware to be able to recharge the batteries.

1.2.1.5 Positioning Unit

Some nodes may need to be mobile to do its task, for example a camera node installed on a robot to patrol an area which in that case may need a positioning mechanism, such implementations are very costly and there are really few such nodes that can be supported in a practical deployment.

1.2.2 Cluster Heads

To reduce the power needed for each sensor node to transmit its data to the base station a cluster head is assigned to each small group of sensors to facilitate the communication, In addition a cluster head may aggregate and partially process the data sent from each node and may reduce the amount of redundant transmissions of the same data from different sensors.

1.2.3 Relay Nodes

Not all the sensor nodes can reach the base station directly; normally hop to hop communication takes place to cope with that. To facilitate this task, the sensor group is divided into smaller groups and a new more powerful sensor node is introduced into each cluster, this new node have a higher power capacity and a larger transmission range in order to be able to reach out to the base station which will be connected to the command center.

1.2.4 Actor Nodes

In some implantations there was a need to have nodes that are capable not just of sensing and monitoring the deployment environment, but also can do certain actions like extinguishing a fire, or call emergency services, or any other intrusive action.

These nodes are very sophisticated when compared to normal sensor nodes, and as a result they are expensive, and when needed, small amount are deployed on a per cluster basis.

1.2.5 Base Stations

Almost in all the deployments a central base station per deployment area is used to collect the data from the different clusters in that area and analyzing them and forwarding them to the command center.

Another function of the base station is to take the commands from the command center and query the sensor nodes, to get the results of the command or force a certain action.

The base station normally doesn't move so it is a stationary node. It has connection to the outer world through different communication media such as satellite, wireless or fixed telecom. It is very powerful in terms of processing and energy capacity.

1.2.6 Command Center

This is the interface to WSN, by which the system user will be able to monitor the system functionality, and through which measurements and environment reports will be accessible to authorized users.

1.3 Applications

WSN have many potential implementations in various aspects of real world applications, they offer a better and closer look into any phenomena that is under study providing higher and controlled data resolutions.

Current sensing technology have the ability to measure almost any parameter around us, starting from temperature, humidity and pressure sensing, going through speed, acoustic, direction, movement and object identification, ending up in providing direct feeds for video and audio streams in addition to providing the exact location using GPS technology.

In this section we provide a brief description of the main groups of applications, with a representative example given on each group.

1.3.1 Military Applications

WSN use in military applications is one of the earliest implementations of this technology in the field. It was used in providing targeting and monitoring functionality in enemy land, and also WSN could provide command and control functionality in addition to surveillance and intelligence information gathering.

A nice example of such an applications is the Boomerang sniper detection system, where a set of acoustic sensors are used to identify the location of the shooter as shown in Figure 1.3 below [13]

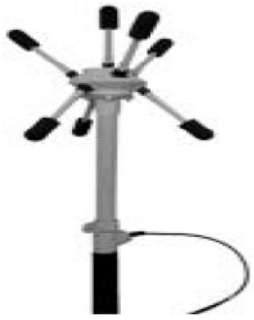


Figure 1.3: Boomerang sniper detection system [13]

1.3.2 Environmental Applications

WSN has been used for detecting and monitoring the movement of animals like birds and fish around the globe. Also it can be used in many other areas like detecting water or air pollution, detecting fires in forests among many other possible applications.

One great application in this area is the early flood detection system, where as shown in Figure 1.4 below, a grid of sensor node is deployed alongside the targeted river, to monitor the water flow in it among other parameters, that make up an input for the flood prediction model, which can alarm local residents of the coming danger [14].

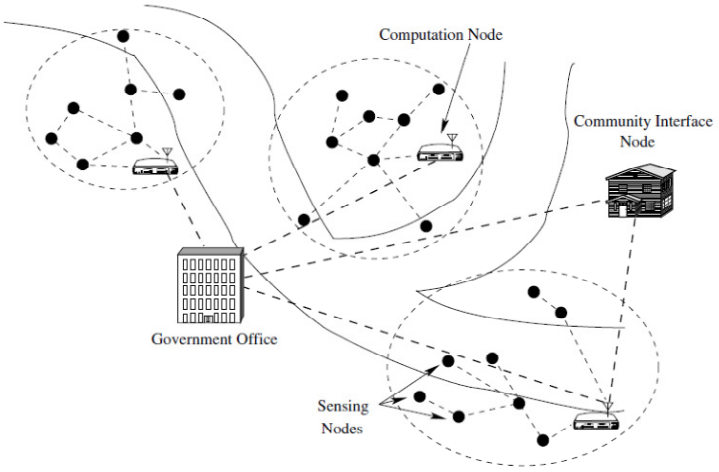


Figure 1.4: Early Flood detection system [14]

1.3.3 Health Care Applications

Large number of health care applications are available today starting from the medical devices that are implanted in the body reaching out to wearable devices that can monitor the patients' biological parameters and then send them to be analyzed by the doctors.

One great application is the Artificial retina, which is being developed, where more than 1000 sensor nodes are to work together to give back the sight for patients' who have lost it because of retina failure as shown in Figure 1.5 [15].

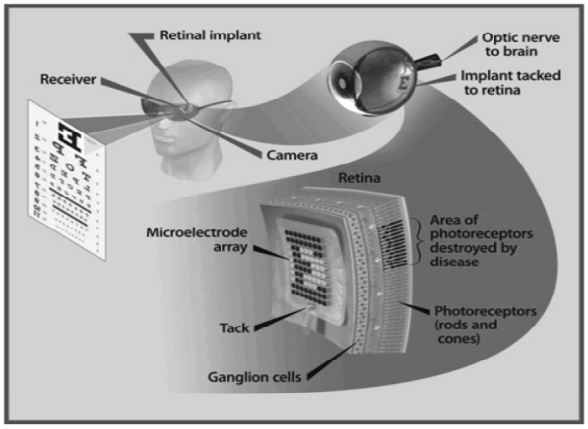


Figure 1.5: Artificial retina [15]

1.3.4 Home applications

Many application of WSN are already in use in our homes today in our DVD's, Computers, mobile phone and is to be implemented in many other appliances like the doors and windows, water and electricity installations in the houses, and above all in security systems.

As an example of home use of WSN, is the water monitoring system that can help us to determine the use of water in each outlet in our homes and not only the total used water as seen in Figure 1.6 below [16].

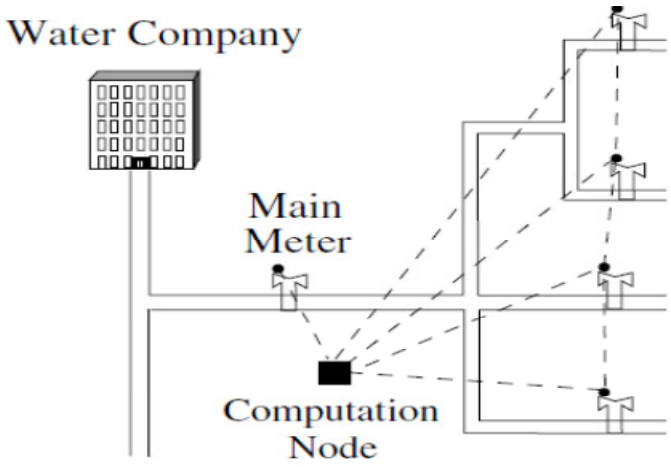


Figure 1.6: Home water monitoring system [16]

1.3.5 Industrial Applications

In all industrial applications there is a need to monitor many parameters, like the work flow, the temperature and pressure in tanks, the quality of the products, and above all to detect problems as early as possible to avoid bigger problems.

One clear example of such an application is structural health monitoring, which is very important to monitor the readiness and safety of certain buildings like bridges and dams, or large production line and heavy duty cranes.

1.4 WSN Constrains and Challenges

The source of all the WSN constrains is the power needed for each node to perform its functionality. As a result nearly all the research and design of anything related to this kind of networks should be power aware [2].

It has been shown that proper placement of different nodes throughout the deployment area has a considerable effect on the network performance since it will increase the network life time by making power consumption very efficient in each node and at the same time achieving best possible coverage and full network connectivity.

Figure 1.7 below outlines the different node placement strategies that have been developed:

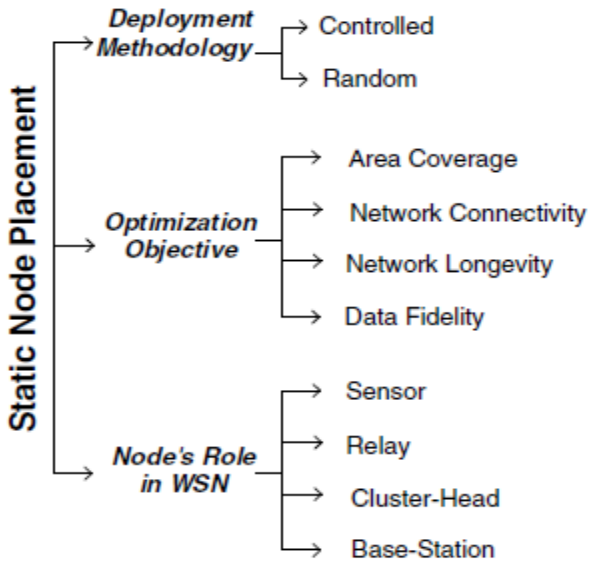


Figure 1.7: Different strategies for node placement [3]

The majority of the applications today use static placement of nodes since to have a dynamic placement would complicate the applications implementation and would require nodes to have a built in mobility and location tracking mechanism. As shown in Figure 1.7 the placement strategies are categorized into three main categories but all of them strive to achieve an efficient network.

1.4.1 Deployment Methodology

There are two methods to deploy the different WSN nodes; controlled deployment is normally used in indoor environments on a node basis criteria. On the other hand the random deployment is used in the rest of the applications.

When the nodes are placed randomly they have to be repositioned to achieve required network performance which will impose a restriction on the nodes to be mobile, another solution may be to add new nodes in the areas with gaps.

1.4.2 Optimization Objective

Ahead of doing any node placement, a certain performance targets are checked and need to be forced during the deployment phase or on a latter repositioning phase, such as:

- Network coverage.
- Latency and delay.
- Redundancy and fault tolerance.
- Energy efficiency.
- Network life time.
- Topology and connectivity.

A typical algorithm would concentrate on one or more of the above points

1.4.3 Nodes role in The Network

Different placement strategies should be used in deployment of the different nodes, since they are different in functionality and numbers.

Sensor nodes constitute the largest percentage of the nodes in the network and normally static placement strategies are used, while in other node types such relays and actors, since there numbers are relatively small static placement is still applicable but dynamic placement strategies give better results and much more flexible topology construction options.

1.5 Scope of Our Work

Node placement problem have been under investigation in the recent years by many researchers around the globe, where different algorithms have been developed. A good survey of these algorithms is provided in following chapter.

Our thesis will focus on extending network lifetime by properly positioning the actor nodes in the deployment area where the main following objectives are met:

- Minimum average delay.
- Increasing network life time by reducing energy consumption.

1.5.1 Methods and Tools

The thesis results are to be verified using simulation environments, no physical or field tests are possible due to lack physical test bids at the university. Suitable simulation tools such as Matlab were used in order to verify our proposed work validity.

1.6 Thesis Problem Statement

The main problem that this thesis will tackle is the optimization of the actor node location within a cluster of sensor nodes to achieve the minimum average delay and to reduce the average energy consumption when any node in the network tries to communicate with the actor.

1.6 Thesis Outline

The thesis is arranged in five chapters including the current chapter arranged in the following manner:

- Introduction chapter, where an overview of the WSN topic is provided with a focus on the applications and constrains, in addition to the thesis scope and framework.
- Related work chapter, where an intensive review of the previous work related to the subject of the thesis is discuss.
- Developed algorithm chapter, where the proposed algorithm is explained.
- Results discussion chapter, where the results of the simulations of the developed algorithm are highlighted.
- Future work and conclusion chapter, where the summary of the main contributions is given in addition to suggestion for future work.

In addition the appendices contain the code used to simulate the algorithms included in this work.

2. Related Work

This chapter provides a survey of the published literature and related works that have explored the topic of this thesis, which will include the topics below:

- ☐ Static node positioning strategies in WSN
- ☐ Node type oriented positioning strategies
- ☐ Dynamic positioning of WSN nodes
- ☐ Conclusion and remarks

This chapter provides a brief literature survey on the node placement strategies in wireless sensor networks that have been developed in recent years, which is classified in the following categories:

- Static placement strategies, which include random and planned positioning of nodes in WSN, their advantages and implementation scenarios, also problems that may arise are highlighted.
- Optimization placement strategies, where nodes are placed based on the optimization targets taking in consideration the coverage and connectivity factors in addition to network life time and their effect on energy efficiency in the network.
- Role based placement strategies of nodes other than sensors, will be discussed for cluster heads and relay nodes.

At the end of the chapter a brief discussion of dynamic placement strategies is also given.

2.1 Static Deployment Methodologies

When deploying a network of wireless sensors in a static fashion two options exist, either controlled or random deployments, depending on the application and the implementation scenarios. For example, in case of the distribution of video sensors for security systems the controlled or planned methodology of placement wins. On the other hand in the case of underwater sensor habitat monitoring random positioning is the suitable and possible solution. Next we briefly discuss both options and related literature.

2.1.1 Planned Static Deployments

When working in closed and well known environment it's better to plan the placement of the sensor nodes since this will result in lower number of deployed nodes, better network topology and above all in an application oriented network.

Planned placement techniques have been proposed and used in many projects such as Active Sensor Network in the university of Sydney [17] which served the purpose of surveillance and facility security. On the other hand the sensor network technology project developed in Intel [18] which focused on providing a factory automation and management functionality in addition to providing a way to do preventive maintenance of the machine prior to failure.

Another application is the structure integrity application where important structures of huge towers, dams, bridges and tunnels are continually monitored to prevent mass damage, example of such application are presented in [19],[20],and [21].

2.1.2 Random Static Deployments

In dangerous open and uncontrolled environments, it's impossible to plan the distribution of sensor nodes; as a result the only feasible option is to randomly scatter the nodes in the area of interest.

Random placement of sensor nodes makes the network topology unpredicted and so the performance of such a network in relaying the sensed data is unreliable. Also the energy consumption throughout the network will not be symmetric as a result of random node density in the area.

In order to enhance the deployment the authors in [22] propose a two layers architecture where relay nodes are added to connect sensors around it to the base station. A disadvantage of this approach was that the nearby relay nodes with large number of surrounding sensors energy levels dropped faster than relay nodes with less number of sensors around them which reduced the whole network lifetime.

To solve this problem, in [23] the authors present a technique in which instead of using one relay node to connect the nearby cluster to the base station, a weighted random distribution was propose to average the sensor density around each relay node, and more relay nodes around the base stations are introduced in order to make the communication between relay nodes and the base station multihop, which save alot of energy instead of long single hop communication as shown in Figure 2.1 below.

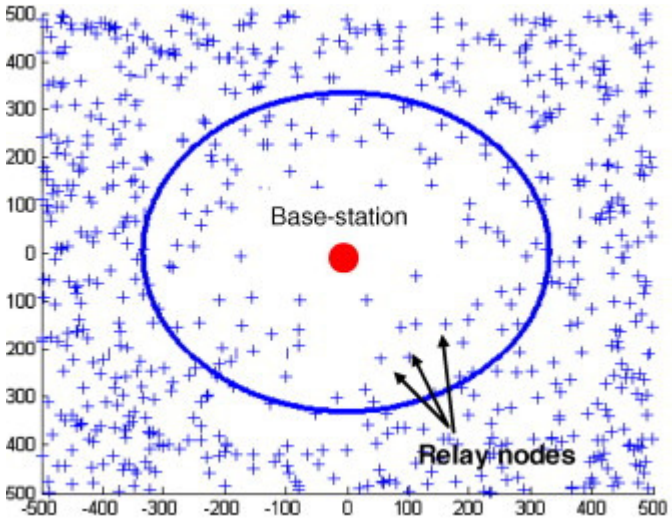


Figure 2.1: The density of relay nodes inside the circle is lower and the connectivity is weaker than outside [22]

2.2 Optimization Objectives

The optimum node placement scenario is not an absolute one, it depends on the optimization parameters upon which the network deployments scenarios have been designed, of course in planned node placements all the objectives can be achieved to some degree while its difficult in random distributions, in this section a literature survey is presented on the work regarding the most important optimization parameters such as:

- Network Coverage
- Network Connectivity
- Network Life time

2.2.1 Network Coverage

The percentage of the area of interest that is covered by the WSN is always needed to be high, especially in monitoring and surveillance applications

The majority of related literature considers the coverage of a single sensor node to be a regular round disk centered at the nodes as described in [24]. The problem with such a model is that it does not precisely provide the required function to model terrain effects or obstacles on the sensor coverage, so its suitable for theoretical studies rather than practical implementations.

On the other hand there are coverage model that are designed to suit specific applications taking in consideration the sensor field of view but still does not count for the terrain as shown in [31].

New breed of models try be closer to reality by working on the concept of least coverage scenario where and object may pass through the area without being detected, this algorithm of measuring the coverage percentage is normally named least exposure coverage as described in [26]

2.2.2 Network Connectivity

The connection among sensor nodes and between them and the base station was considered earlier a solved problem. As a result of the assumption that the transmission range of sensor nodes is long enough to make the network fully connected at all times.

In [27] the authors assumed the transmission range of a node to be limited to its sensing range. To keep the network connected as long as it is a 2D network, they proposed to make the distance between the sensor nodes less than twice the transmission range of a single node, but this method created congestions in the path towards the base station.

In [28] the authors build a k-connected network to provide different paths to the base station while at the same time make the network much more fault resilient. Other authors look at the problem from a general perspective to solve congestions and bandwidth related issues such as [29].

2.2.3 Network Life Time

The most important objective that a network designer is concerned with is the lifetime of the network, that is when the network will stop to function as a result of nodes die out because of energy depletion.

As described in [22] the variation of sensor nodes density in the area results in making the nodes around the base station and the nodes at the edge of the deployment area depleting their energy in a shorter time than other nodes and so killing the network.

2.3 Node Type Oriented Positioning Strategies

The previous discussion considered all the nodes in the WSN network to be of the same type, which is far from the real application scenario. Typically the roles are functionally distributed among different layers of nodes resulting in a multitier network with different layers as shown in Figure 2.2 below:

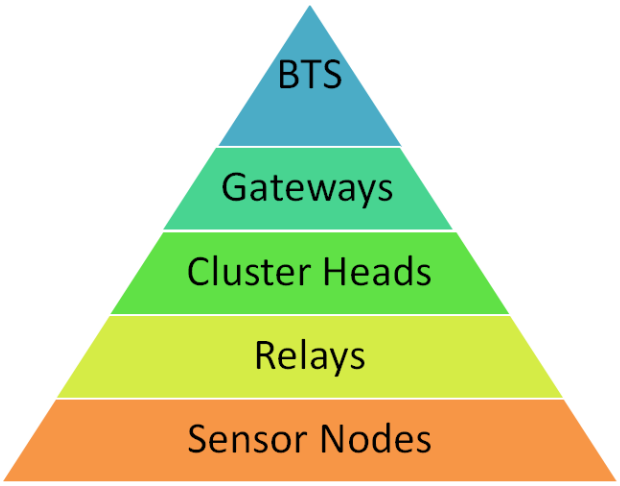


Figure 2.2: Hierarchal structure of a WSN network

Where the relay and aggregator nodes function as a long transmission backbone that collects traffic and forwards it towards the cluster heads, which in turn concentrates the relayed traffic towards the base station which is the control center of the network.

Instead of focusing on the whole WSN nodes, researchers moved towards planning the locations of certain key nodes that play as a backbone and data concentration for sensor traffic.

This approach resulted in better performance of the overall network but also introduced some new challenges such as single points of failure and faster rate of energy depletion for the backbone nodes which may result in dissecting the network into separated islands.

2.3.1 Aggregation Relay Nodes Repositioning

Relay nodes allow the WSN network topology to be flexibly shaped and controlled, through building long haul links between relay nodes to facilitate communication between all the nodes in the network. Relay nodes may be normal sensor nodes in flat networks, or they can be special nodes with long transmission range and large energy reserves in layered WSN.

Figure 2.3 below illustrates how a relay node plays a central role in supplying a communication hub between the sensors attached to it and the rest of the network.

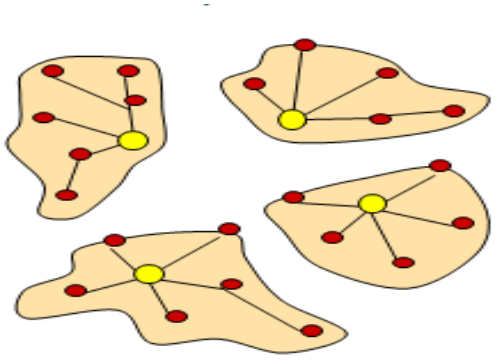


Figure 2.3: Sensor and relay nodes

Early research on the topic suggested a two layer network as shown above in Figure 2.3; to solve the problem of higher energy consumption at the relay nodes the researchers in [34] suggested either adding more nodes or implementing an energy monitoring on the actuator nodes to increase the efficiency.

Following research tried to find the least number of actuator nodes that are needed to have a connected network of several degrees k=1, and k=2, in order to make the task possible they divided the network into clusters of small cells, each would contain an relay node, as in [33]

On the other hand in order to increase the life time of the network and enhance the connectivity researchers in [35] suggested a three layer WSN network where two levels of relays are used to share the network communication load and act as backup to each other.

2.3.2 Cluster Heads Repositioning

In order to build better WSN topologies the network must be divided into clusters where cluster heads play the main role in accomplishing this task. Cluster heads are powerful sensor node with long transmission range and high processing capacities. It utilizes the relay nodes to control the WSN network to achieve the following functions:

- Aggregate and disseminate sensors traffic.
- Control sensor node operation within the cluster and coordinate their operation.
- Maintains communication topology and routing tables.
- Forwards traffic to the base station through multihop communication with other cluster heads.

It is very important to position cluster heads in a studied manner since it will be reflected on the performance and optimization parameters of the network as the majority of researchers and publication confirm.

Figure 2.4 below shows a complete WSN network with sensor nodes in red, the relay nodes in yellow and the cluster heads in green where all sensor traffic is streamed towards the base station where the control center is located.

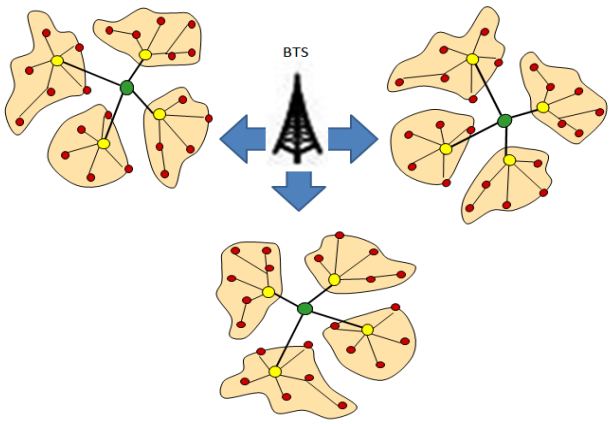


Figure 2.4 Cluster heads Role in the WSN network

The research in cluster heads positioning is split in, either placing the cluster heads in the network after the sensor cluster have already been formed, or before the sensor clustering process takes place.

As for cluster heads placement after sensor clustering is done, the cluster heads positioning within each cluster is done separately and irrelevant of other clusters in the network, where the target is to put the cluster heads in a location where the latency and energy consumption are kept it to a minimum which will result in a longer life time for the WSN network.

Authors in [36] used k-mean clustering to minimize the direct distance to the cluster heads from each sensor in the cluster, while researchers in [37] used genetic algorithms and neural networks to find a location around the center of the whole cluster that will minimize the energy consumption and latency.

In the case of placing the cluster heads prior to clustering the sensor nodes, the problem is more difficult because the placement choices are bigger and the amount of cluster heads nodes needed to cover the deployment area has to be determined upon the performance required in terms of network coverage, latency, energy usage and network life time.

A method to tackle the pre-clustering to cluster heads placement was developed in [38] base on minimization function that would minimize the maximum energy used by a single sensor node in the WSN network while keeping overall energy consumption to a minimum, after that a routing algorithm is used to build the clusters (cluster heads - Sensor associations).

Another method to solve the problem is by finding the position that will be at a minimum distance from all sensors within a cluster, the method suggested in [39] achieves the target by putting the cluster heads at the center of a circle that has the furthest three sensor node on its circumference, this method allows some kind of control on the size of the cluster around the cluster heads nodes.

2.4 Dynamic Positioning of WSN Nodes

Wireless sensor network nodes are venerable to damage by exterior factors, or they may die out after their energy has been depleted. Therefore a new breed of node replacement algorithm is needed in order to restore the WSN network to its desired operational state once some of the sensors of any higher layer node stop working or even vanish.

A new area called topology maintenance and control is being formulated to control the phase of building the WSN network taking in consideration providing fall back scenarios and alternative communication paths with redundant nodes, which will solve node failure issues with new network topologies when needed.

In order to be able to achieve those targets DEMA must have the capability of detecting network failures to be able to react to them.

The area of wireless sensor networks is a very hot topic, that is rich of challenges in many areas starting from the very basic tasks and building blocks of WSN node itself utilizing the cutting edge ultra low power electronics, going through advance sensor equipments and efficient communication protocols devise for WSN platforms.

This chapter supplied a comprehensive brief literature survey on different WSN nodes positioning strategies and related algorithms that have developed in recent years to make the WSN more responsive and fault tolerant, also some algorithms worked on enhancing the coverage and increasing network operational life time.

The related work and DEMA were classified into two main categories:

1. Static WSN node positioning which meant that the location of nodes in the network can't be change once they are rolled out in the targeted area

    - ☐ Random node placement, where the nodes are randomly scattered in the area without any prior planning of the position, it is the only possible solution in huge open environments.

    - ☐ Planned node placement, where the nodes locations are studied carefully before they are deployed which results in a better network performance, but is only suitable for indoor and controlled environment.

2. Dynamic WSN node positioning, where all the nodes or a type of them are allowed to change their position after the initial deployment, which helps alot in solving problems that may arise during the operation of network, resulting in better network performance.

Those strategies could be applied on different levels in the network:

- All the nodes in the network (Sensor nodes, Relay nodes, Cluster heads and actuators).
- Only for a certain type of nodes, this is the most common scenario.

3. Delay and Energy Saving by Actuator Positioning (DEMA) Formulation

This chapter provides a thorough discussion of DEMA and its basic blocks and functions.

- ☐ Network bootstrapping and adjacencies formulation
- ☐ Solving all pairs shortest path problem
- ☐ Formulating the energy consumption model
- ☐ Conclusion with DEMA pseudo code

This chapter presents the algorithm we developed in a comprehensive manner; the algorithm aims to locate the actors in a position within a sensor network cluster to achieve the following optimization objectives:

- Minimum average delay in communicating data between any sensor node in the cluster and the controlling actor.
- Reduced average energy consumption on the network level by minimizing the average number of hops between any sensor node and the controlling actor

In order to achieve the above targets the algorithm goes into many stages that are briefly shown in Figure 3.1 below, and will be thoroughly discussed in this chapter:

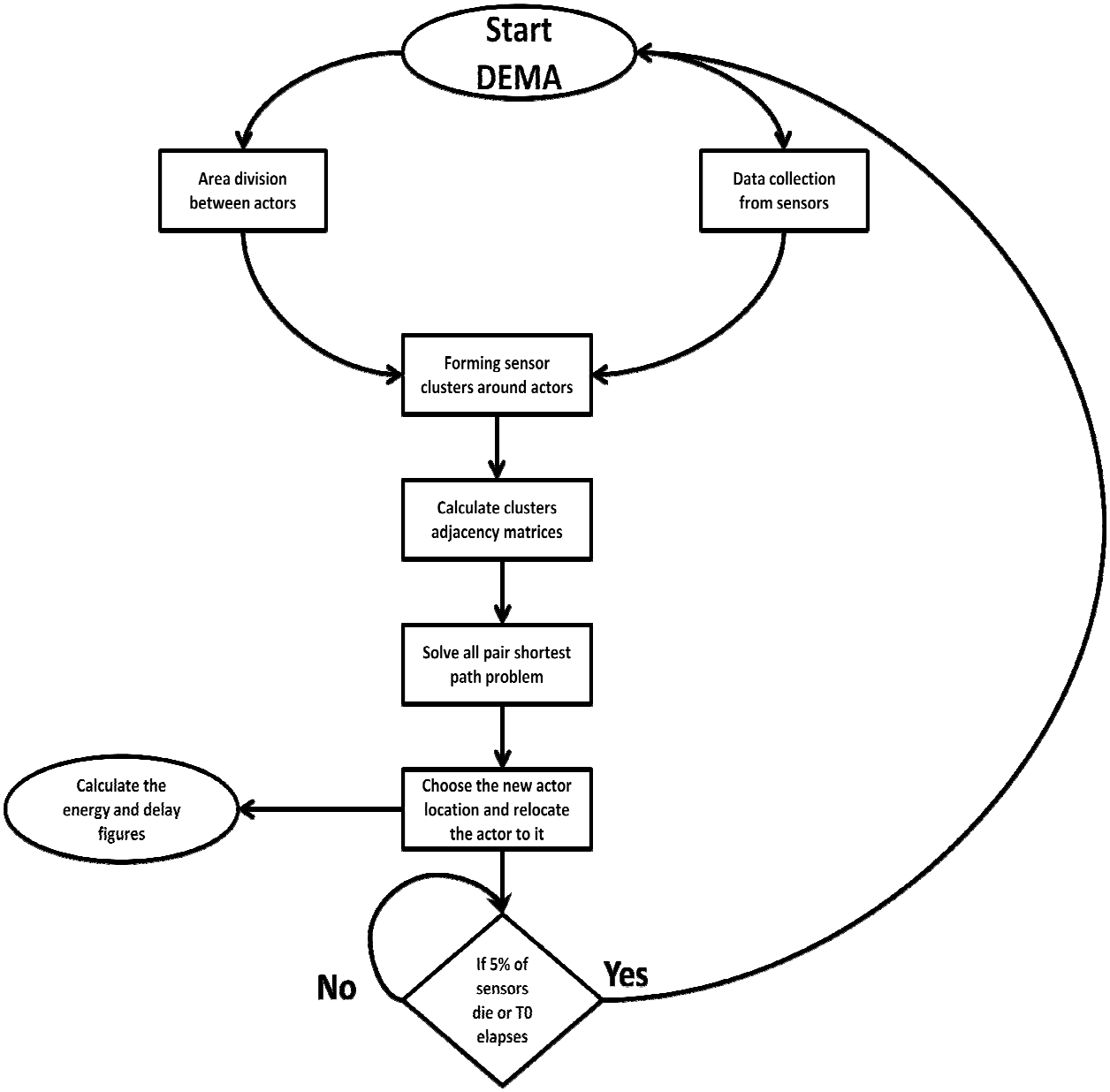


Figure 3.1: DEMA flowchart

3.1 Actuator Level Network Bootstarp

The following assumptions are considered to start this stage:

- The actors are randomly deployed and can locate themselves in the deployment area.
- The deployment area boundaries are known.
- The actors are mobile
- The actors transmission range is large enough to cover deployment area

The actor nodes will send with each other their current locations in the deployment area and then do the area division algorithm described below in order to determine their new location

3.1.1 Area Division Algorithm

As assumed above, the actors are scattered in a random fashion across the area of interest, as a first step in relocating them, an algorithm is needed to determine the best nominal location of the actors, the used algorithm in this work depends on the binary tree algorithm in computer science which is normally used for searching the databases and building classified data sets.

A modified binary tree algorithm is used in this work to divide the main area into equal subareas if the number of actors is even and to n-1 equal subareas and one bigger subarea if the actors number is odd as shown in Figure 3.2 below.

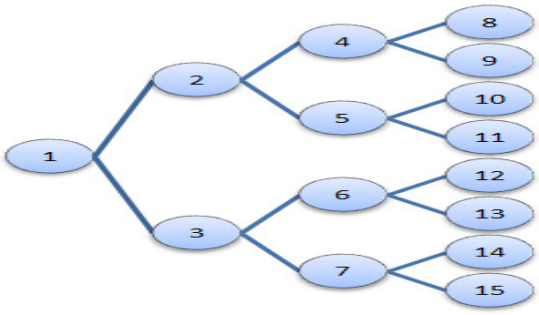


Figure 3.2: Binary Tree algorithm

The algorithm is executed as follows:

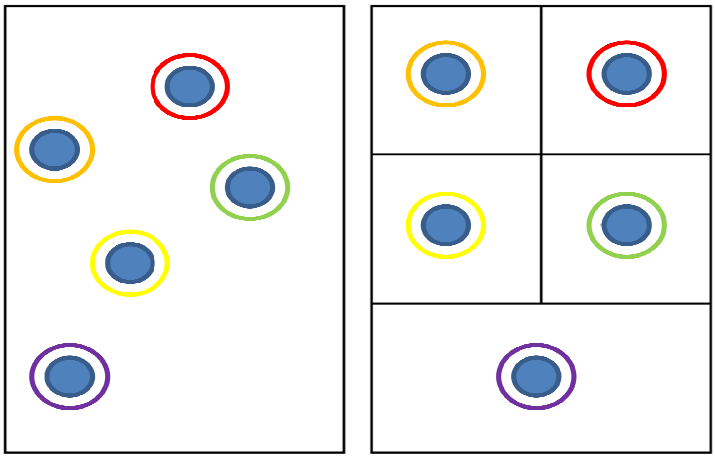
- The area is assumed to be rectangular to ease calculation, and its boundaries are provided as input to the algorithm along with number of actors.
- The algorithm checks the number of actors, if only one actor is used then no need to follow on and the actor is relocated to the center of the deployment area,
- If more than one actor is used, it recursively divides the main area to create a number of subareas matching the number of used actors as shown in the pseudo code in Figure 3.3.
- Once the subareas are calculated with their corresponding centers, each actor calculates its Euclidian distance towards each subarea center and mobilizes itself towards the nearest subarea center as explained below.

This algorithm is executed separately on each actor; its results are the same on each actor since its inputs are the same. The algorithm is not conflict free, and to solve this issue each actor broadcasts the location it's going to in order to direct other actors away from it, in the case where two or more actors have the same Euclidean distance from a center of a subarea..

```
% As an input to the algorithm the number of actors N and the
% area boundaries Area are given.
Subarea[N,Area];
If N = 1
Actorlocation = Center(Area)
Else
Find(Area,area1,area2)
Area1= Subarea[(N/2),Area]
Area2= Subarea[(N/2),Area]
      End
% The output of the above step is a set of subareas and centers
%, next each actor calculates the closest subarea and moves to it
For i := 1 to N
For j := 1 to N
Actortocenter[i,j] =sqrt ((X_A − X_C)^2 + (Y_A − Y_C)^2)
End
% we take the minimum distance between any actor and the
% subarea centers sand move the actor to it.
For i := 1 to N
Locationofactor [i] = Center with Min(actortocenter[i,:])
End
% Final step is to move the Actor to the selected center
Move Actor[i]to locationofactor [i]
End
```

Figure 3.3: Actor area division algorithm

Next a sample run of the above pseudo code is shown in Figure 3.4, where five actors are to be deployed, as it can be seen, the result was four equal subareas and one bigger subarea, also each actor was moved to the nearest subarea.



(a) **(b)**

Figure 3.4: An example of the division algorithm, where (a) shows the actor locations befor clustering, while (b) is after clustering

After the algorithm in Figure 3.3 is executed, each actor knows where to go. Then each actor relocates itself to the new location which is the center of the subarea where it will control the sensors within.

3.2 Data Collection and Sensor Network Bootstrap

The following assumptions are considered in this stage:

- The sensor nodes are aware of their locations.
- The nodes can communicate with each other and with actors by flooding their profile.

After the actor is mobilized to the center of the subarea that it will manage, the sensors in that area will communicate between them and with actor, sending their profiles and details which will include the data shown in Table 3.1 below.

Node Model	
Node Id	Should be unique
Node Position	The x and y coordinated for the simulation purposes or the latitude longitude GPS obtained data
Energy Level	Used for simulating energy efficiency of different topologies.
Neighbors List at Max. Transmit. Power	Used for decision making in building the topology, where all the sensor nodes will be transmitting at maximum power
Neighbors List at Tuned Transmit. Power	Used for decision making in building the topology, where the sensor nodes will be transmitting at needed power to reach neighbors'
Used Path to Actuator	Used when the topology rotation is performed to assign new path for the specific node to reach the actuator

Table 3.1: Node profile and details

3.3 Forming the Adjacency Matrix

In order to construct the adjacency matrix of the whole network after each actor has been mobilized to the center of the corresponding subarea, the locations of each node (Both sensors and actors) in the network are collected and arranged in a vector with their respective identities as shown in Figure 3.5 below where the first column represents the node ID, the other two columns represent the XY coordinates of the node in 2D plane:

$$\begin{pmatrix} Node01 & x01 & y01 \\ Node02 & x02 & y02 \\ Node03 & x03 & y03 \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ NodeN & xN & yN \end{pmatrix}$$

Figure 3.5: Nodes location vector

The vector in Figure 3.5 is assumed to be available at each node in the network, in order to start preparing the inputs to run DEMA. The inputs are prepared in two phases, first each sensors need to associate itself with a certain actor to build the network clusters. Once this step is done each sensor forms adjacencies with nearby sensors to reach out to the actor controlling the cluster as a second phase.

3.3.1 Forming Clusters Phase

After the actors are at the centers of their subareas the sensors in the whole deployment area start calculating the Euclidian distance between their locations and the location of all the actors in the network according to the equation below:

$$D_{as} = \sqrt{(X_a - X_s)^2 + (Y_a - Y_s)^2} \qquad \textbf{3.1}$$

Where:
(X_a, Y_a) is the actor location
(X_s, Y_s) is the sensor location
D_{as} is the 2D Euclidian distance between the sensor node and the actor.
The result is stored in a K x N matrix where N is the number of sensor nodes in the network and K is the number of actors as shown in Figure 3.6 below

$$\begin{pmatrix} D_{11} & D_{12} & D_{13} & : & : & : & : & D_{1N} \\ D_{21} & D_{22} & D_{23} & : & : & : & : & D_{2N} \\ : & : & : & : & : & : & : & : \\ : & : & : & : & : & : & : & : \\ D_{K1} & D_{K2} & D_{K3} & : & : & : & : & D_{KN} \end{pmatrix}$$

Figure 3.6: 2D Euclidian distance between all WSN nodes and actors

From Figure 3.6 above each sensor finds the nearest actor to it by taking the minimum entry in its column, and then associates itself to that actor, which results in clustering the network into a number of groups equal to the number of actors.

3.3.2 Forming Clusters Adjacency Matrix

In this phase each cluster does a separate calculation of the distances between the nodes in the cluster (Sensors plus the actor node), again using the Euclidian distance equation 3.1 we used previously. The result is formulated into a (n+1) x (n+1) squared matrix where n+1 is the number of the sensor nodes in the cluster plus one actor as shown in Figure 3.7**:**

$$\begin{pmatrix} D_{11} & D_{12} & D_{13} & : & : & : & : & D_{1(n+1)} \\ D_{21} & D_{22} & D_{23} & : & : & : & : & D_{2(n+1)} \\ : & : & : & : & : & : & : & : \\ : & : & : & : & : & : & : & : \\ : & : & : & : & : & : & : & : \\ : & : & : & : & : & : & : & : \\ : & : & : & : & : & : & : & : \\ D_{(n+1)1} & D_{(n+1)2} & D_{(n+1)3} & : & : & : & : & D_{(n+1)(n+1)} \end{pmatrix}$$

Figure 3.7: 2D Euclidian distance between all cluster nodes per cluster

So Figure 3.7 represents the direct distance between any two nodes in the cluster. From it the adjacency matrix is built, first Full Power adjacency matrix is formulated by setting the nodes transceivers to work at full power which maximizes the distance over which the node can communicate in order to make the matrix cover the largest number of possible paths connecting various network parts.

To build the full power adjacency matrix a critical transmission (CTR) range is defined as a threshold over which a node can't make connection and below it communication is possible as shown below in 3.2:

$$FAdj\left(D_{(n+1)(n+1)}\right) = \begin{cases} 1, D_{(n+1)(n+1)} < CTR \\ 0, D_{(n+1)(n+1)} \geq CTR \end{cases} \qquad \textbf{3.2}$$

Where:
$FPAdj()$ Is the (n+1) x (n+1) adjacency matrix of a certain cluster, and $D_{(n+1)(n+1)}$ is the cluster distance matrix, and CTR is node critical transmission range at full power.

Figure 3.8 is an example of how such a matrix would look like, from the matrix it can be said that sensor node 1 has a direct link with sensor node 2 since D_{12} is less than CTR, but none of them have a direct link with sensor node 4 since D_{14} is larger than CTR :

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Figure 3.8: Sample cluster adjacency matrix

Of course if the transceivers used power is less than the maximum, fewer connections will be possible and so less ones in the above matrix which will be no longer a full power adjacency matrix. The above procedure is summarized in pseudo code below in Figure 3.9, and will be repeated on a per cluster basis, and its outputs will be fed to the next stage in the algorithm

```
1. % As an input to this stage an n x 3 vector is provided where its entries
2. % are as explained before are a triplet (Node id,X location, Y location),
3. % and n is the number of nodes, CTR is the critical transmission range.
4. Nodelocation[n,3] = (Noden,Xn,Yn);
5. Distacemtrix[i,j];
6. For i := 1 to n
7. For j := 1 to n
8. Distacemtrix[i,j] =sqrt ((X_i − X_j)^2 + (Y_i − Y_j)^2)
9. Adjacency[i,j];
10.    For i := 1 to n
11.    For j := 1 to n
12. Adjacency[i,j] = { 1, DistaceMtrix[I,j] < CTR
                     { 0, DistaceMtrix[I,j] ≥ CTR
13.    % The output of the algorithm is an n x n matrix where its
14.    % entries path(i,j)= 1 or 0,depending whether we have a  path or not
```

Figure 3.9: Cluster adjacency matrix formulation

3.4 Solving All Pair Shortest Path Problem

The need to find the shortest path between two locations or a set of minimal cost paths rises in a lot of applications; the most famous problem is the travelling sales man problem where the sales man needs to visit a network of locations without passing the same way twice.

In graph theory this is translated into a set of vertices connected by edges which represent the paths, those edges can be directional or unidirectional, weighted or not weighted, also the graph could be fully connected or not.

Many algorithms have been developed to solve different facets of shortest path optimization problem, which fall into one of the following categories:

- The single pair shortest path problem: where only two vertices are studied and the path connecting them is considered.
- The single source shortest path problem: where the interest is to find the shortest paths to all vertices starting from a certain vertex.
- The single destination shortest path problem: where the objective is find the minimal path from each vertex to a certain destination vertex.
- The all pairs shortest path problem: where the shortest path is to be determined for each pair of vertices in the graph.

In our work, for optimizing the delay and energy in wireless sensor network the last category of shortest path algorithms offers the required optimal way of processing the topology of the network to be further optimized.

The most famous algorithms of the all pairs shortest path category are the Floyd–Warshall algorithm and the Johnson's algorithm. Below a brief description of both is provided.

3.4.1 Floyd–Warshall Algorithm [30]

The algorithm is very simple and used to solve the all pair shortest path problem for dense graphs; it depends on the triangle ruler that can be stated as:

$$\text{Path}(i,j) = \text{minimum}\ (\ \text{Distance}(i,j);\ \text{Distance}(i,k) + \text{Distance}(k,j))$$

$$\text{So Path}(i,j) = \text{Distance}(i,j) \text{ if } \text{Distance}(i,j) < \text{Distance}(i,k) + \text{Distance}(k,j)$$

$$\text{Or Path}(i,j) = (\text{Distance}(i,k) + \text{Distance}(k,j)) \text{ if } \text{Distance}(i,j) > \text{Distance}(i,k) + \text{Distance}(k,j)$$

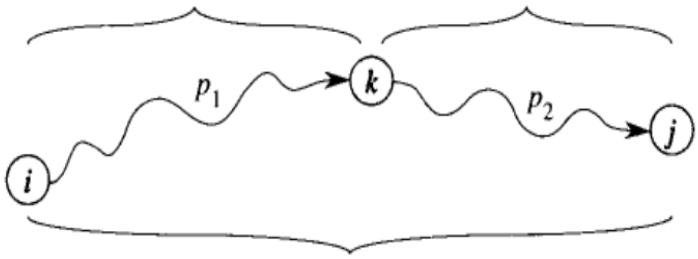


Figure 3.10: Triangle rule where the direct path between i and j is compared with path from i to j through k [30]

From the statements above and the Figure 3.10 shown above, the algorithm starts by getting the adjacency matrix of a graph represented by a square symmetric bidirectional matrix, then execute the above mentioned triangle rule as shown in the next pseudo code :

```
1. % As an input to the algorithm an n x n adjacency matrix is provided,
2. % where its entries are 1 if there is an edge(i.j)and 0 otherwise
3. Adjacency[i,j];
4. Floydwarshall (Adjacency)
5. For k := 1 to n
6. For i := 1 to n
7. For j := 1 to n
8. Path[i, j] = min (Adjacency [i,j], Adjacency [i,k]+ Adjacency [k,j] );
9. % The output of the algorithm is an n x n matrix where its
10.     % entries path(i,j)= the number of hops between nodes i & j.
```

Figure 3.11: Floyd-Warshall algorithm

So we have three for loops to achieve the goal of the algorithm, it is obvious that we have $\Theta(|N|^3)$ complexity which proves that the algorithm is very efficient since we have N^2 edges in the graph to be checked for N sensor nodes. When applying the Floyd-Warshall algorithm on the adjacency matrix in Figure 3.8, the output will look like the Figure 3.12 where each entry represents the distance in hops from each node to any other node in the cluster, for example the distance between node 3 and node 1 is two hops.

$$\begin{pmatrix} 0 & 1 & 2 & 3 & 1 & 2 & 2 & 2 & 1 & 2 \\ 1 & 0 & 1 & 2 & 2 & 1 & 2 & 1 & 2 & 1 \\ 2 & 1 & 0 & 1 & 3 & 2 & 1 & 2 & 1 & 2 \\ 2 & 1 & 2 & 0 & 2 & 1 & 2 & 2 & 3 & 1 \\ 1 & 2 & 3 & 2 & 0 & 2 & 2 & 1 & 2 & 1 \\ 2 & 1 & 2 & 1 & 2 & 0 & 3 & 1 & 3 & 2 \\ 2 & 2 & 1 & 2 & 2 & 3 & 0 & 3 & 1 & 1 \\ 2 & 1 & 2 & 2 & 1 & 1 & 3 & 0 & 3 & 2 \\ 1 & 2 & 1 & 2 & 2 & 3 & 1 & 3 & 0 & 2 \\ 2 & 1 & 2 & 1 & 1 & 2 & 1 & 2 & 2 & 0 \end{pmatrix}$$

Figure 3.12: Floyd-Warshall algorithm output

The algorithm is very fast and doesn't consume alot of memory.

3.4.2 Johnson's Algorithm [30]

This algorithm is used to solve the all pair shortest path problem for sparse directional graphs; the algorithm is practical implementation of two other algorithms, Bellman–Ford algorithm and Dijkstra's algorithm. It goes through the following steps:

- A new node is introduced which is connected to all vertices in the graph with zero weight.
- Next the Bellman–Ford algorithm is executed to find out the shortest path from the introduced node to all the other vertices, to check if there are any negative cycles, if any of them exists, the processing is terminated.
- Once Bellman–Ford algorithm terminates successfully, the weights resulting from it are assigned to the original graph (without the newly introduced node and its paths).
- At the end Dijkstra's algorithm is executed on the resulting weighted graph from the Bellman–Ford algorithm, to find the all pair shortest path problem solution.

3.4.3 Comparison of Floyd–Warshall and Johnson's Algorithms

As a starting remark, both algorithms do the job of solving the all shortest path problem, below the characteristics of each algorithm are highlighted:

Floyd–Warshall algorithm

- This algorithm can be applied for dense graphs and also for sparse graphs.
- It can deal with negative weighted edges and negative cycles.
- It utilizes the triangle rule.
- $\Theta(N^3)$ complexity order.

Johnson's algorithm

- This algorithm can only be applied for sparse graphs.
- The graph shouldn't have any negative edges of cycles.
- Based on merging Bellman–Ford and Dijkstra's algorithms.
- $\Theta(n^2 * \log n + N*e\log n)$ complexity order

From the above shown properties of each algorithm, it's clear that the more general and suitable one for solving the all pair shortest path problem in wireless sensor networks is Floyd–Warshall algorithm, but in terms of efficiency Johnson's algorithm is better where it may be applied.

In developing the proposed algorithm Floyd–Warshall is considered due to its generality and flexibility.

3.5 Choosing the New Actor Location

As an input to this step, the adjacency matrix and the all pair shortest path matrix should be ready as shown below in Figure 3.13:

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}$$

(a)

$$\begin{pmatrix} 0 & 1 & 2 & 3 & 1 & 2 & 2 & 2 & 1 & 2 \\ 1 & 0 & 1 & 2 & 2 & 1 & 2 & 1 & 2 & 1 \\ 2 & 1 & 0 & 1 & 3 & 2 & 1 & 2 & 1 & 2 \\ 2 & 1 & 2 & 0 & 2 & 1 & 2 & 2 & 3 & 1 \\ 1 & 2 & 3 & 2 & 0 & 2 & 2 & 1 & 2 & 1 \\ 2 & 1 & 2 & 1 & 2 & 0 & 3 & 1 & 3 & 2 \\ 2 & 2 & 1 & 2 & 2 & 3 & 0 & 3 & 1 & 1 \\ 2 & 1 & 2 & 2 & 1 & 1 & 3 & 0 & 3 & 2 \\ 1 & 2 & 1 & 2 & 2 & 3 & 1 & 3 & 0 & 2 \\ 2 & 1 & 2 & 1 & 1 & 2 & 1 & 2 & 2 & 0 \end{pmatrix}$$

(b)

Figure 3.13: (a) Sample adjacency matrix, (b) Floyd-Warshall output when (a) is the input

In order to determine the best location to move the actor to, the following steps should be done in the all pair shortest path matrix. :

- Calculate the average of each raw, which will indicate the average distance each sensor node in the network will go through in order to reach the sensor represented by that raw.
- Calculate the number of ones and twos in each raw, which will show how many nodes in the network rely on the sensor represented by that raw, to connect to other nodes in the network.

The pseudo code below shown in Figure 3.14 illustrates how the calculations are done to locate the best position to put the actor in:

```
1. % As an input to the algorithm an n x n all pair shortest
2. % path matrix. Is provided.
3. Path[i, j];
4. For i := 1 to n
5. Avg{i}= Average path[i, :];
6. Number of Ones[i] = Ones path[i, :];
7. Number of Tows[i]= Tows path[i, :];
8. End
9. Avg Location = the sensor location with Min (Avg{i});
10.   Ones Location=sensor location with Highest (Number of Ones[i] {i});
11.   Twos Location= sensor location with Min (Number of
12.   Tows{i});
13.   New Actor Location = (Avg Location + Ones Location + Twos Location )/3;
14.   Move the actor to the new location;
```

Figure 3.14: New actor location calculation

If we implement the above procedure using the previous all pair shortest path the following results are obtained:

Node	Average	Ones	Tows
Sensor01	1.6	3	5
Sensor02	1.3	5	4
Sensor03	1.5	4	4
Sensor04	1.6	3	5
Sensor05	1.6	3	5
Sensor06	1.7	3	4
Sensor07	1.7	3	4
Sensor08	1.7	3	4
Sensor09	1.7	3	4
Sensor10	1.4	4	5

Table 3.2: Parameters for selecting new Location for the actor

From the table above, Sensor02 has the least average path length towards other nodes, and it also has the largest number of nodes one hop away from it, but when it comes to the number of twos four sensors have the same number of nodes two hops away, the algorithms considers the first one of them and discards the rest.

Then the new actor locations will be close to Sensor02 but not at its same locations, which distinguishes the algorithm from all the published work where only sensor locations were considered and not other locations like in our case.

By following up this algorithm, the delay in the whole network is minimized, but also the actor will be placed in a location where the highest percentage of nodes are located in, by doing this the energy required to transmit or receive is kept to a minimum because the distances towards the actor will be small, when compared to putting the actor in the center of the cluster in case where the sensor distribution may random or not uniform.

3.6 Power Model and Evaluating Energy Efficiency

Developing a model for energy consumption from scratch is not possible in our case since no hardware (test bid platform) is available to test or build the algorithm on, so it was better to formulate a model based on published literature on this topic. Harbin University in china was adopted [32], below is a brief illustration of the model is given. After that refined portion of it is formulated to be applied for the purposes of our work.

When looking at the wireless sensor node from a power consumption point of view, three main categories arise:

- SE - Sensors subsystem energy consumption
- PE - Processing subsystem required energy
- TE - Transceivers subsystem energy consumptions

Of course there are other elements and circuits that can also consume some energy such as the control circuits and storage devices among other things, but still the total energy consumption by those modules is a small percentage of the whole system as shown below in Figure 3.15.

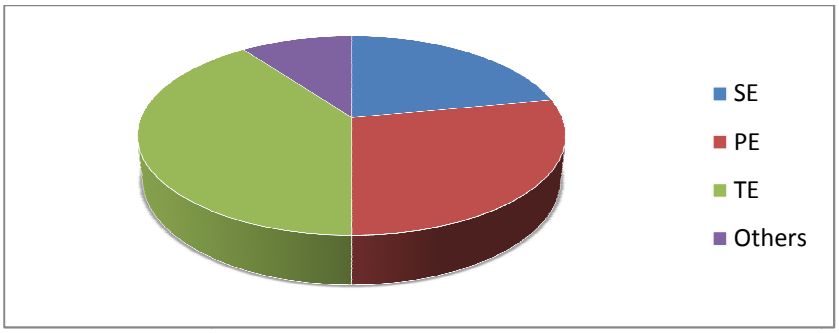


Figure 3.15: Energy consumption per WSN node Module [32]

3.6.1 Sensors Subsystem Energy Consumption

Different types of sensors could be mounted on board a wireless sensor node depending on the application ranging from climate sensors to audio and video cameras, ending up with a wide range of chemical sensors and more. Figure 3.16 below shows how the energy is consumed by a Dallas digital temperature sensor DS18B20:

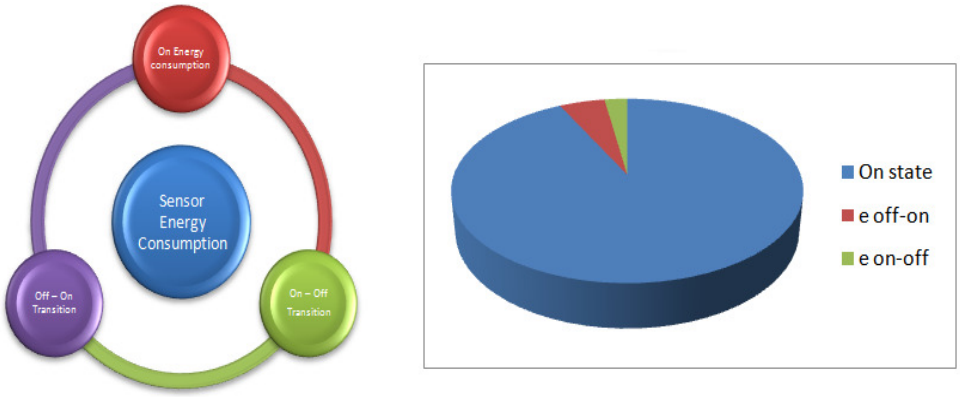


Figure 3.16: Energy consumption in the sensor subsystem [32]

The energy consumption of such sensors can be simply modeled as if they were an electrical resistive load as shown in 3.3 and 3.4 below:

$$E_{sensor} = E_{on-off} + E_{off-on} + E_{sensor-run} \qquad \textbf{3.3}$$

$$E_{sensor} = N\left(e_{on-off} + e_{off-on} + V_s I_s T_s\right) \qquad \textbf{3.4}$$

Where the E_{sensor} is total energy consumption by the sensor, and Eon-off and Eoff-on is the energy required by the sensor to power on and to turn off respectively, while Vs and Is are the consumed voltage and current at the while working, and Ts is the time the sensor is up and doing measurements, finally N is the number of time the whole cycle is repeated

In Table 3.3 below typical values from field trials have been obtained for various types of sensors excluding cameras:

Parameter	Quantity	
Is	1×10^{-3}	A
Vs	5	V
Conversion time	0.75	S
E off-on	2×10^{-4}	J
E on-off	1×10^{-4}	J

Table 3.3: Typical values for sensor power usage [32]

If the above values are plugged into the energy consumption equation of the sensor for a single measuring or sensing period:

$$E_{sensor} = 4 \times 10^{-3} \text{ Joules}$$

3.6.2 Processing Subsystem Energy Consumption

Typically each node in the network has a processing capabilities that enables it to interpret the sensors input and form them into a digital sequences that can be transmitted towards the control interface through the actors. In addition some nodes that act like a hop to communicate other nodes data to actors, which needs some processing and coordination.

As a result, the nodes processing subsystem energy consumption varies depending on the role of the node in the network, so nodes at the further edges of the network consume less energy in terms of processing requirements since they only process their sensors data, but as we go deeper into the network towards the actors more processing is required to handle other nodes data, which results in a higher energy consumption to process the incoming streams from the surrounding nodes.

As shown in Figure 3.17 below the processing modules of Intel Strong ARM SA-1100 has three states, Sleeping mode where the minimum power is consume, and Idle mode where the energy consumption is higher but still low at this mode the node can quickly start processing incoming data, finally the Run mode where the node is actually processing data, the run mode consume the highest amount of energy compared to the other two modes.

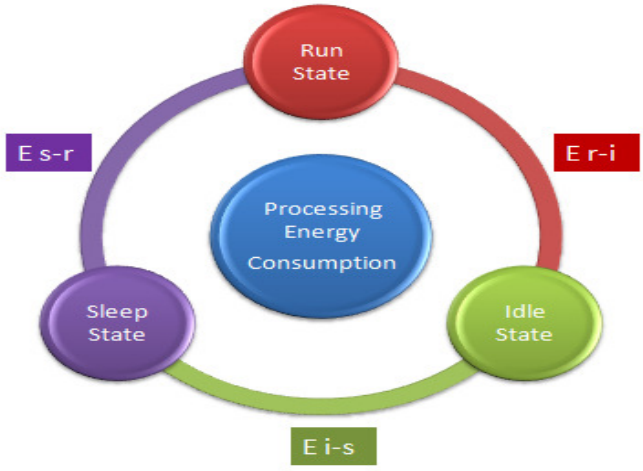


Figure 3.17: Processing subsystem operation modes and transoms [32]

In addition to the three states of operation there are five possible transitions between the different states, as will be shown below, those transition also add up to the energy consumption.

The overall consumption of the Processing function within a WSN node can be given by the following equations:

$$E_{cpu} = E_{cpu-state} + E_{cpu-change} \quad \textbf{3.5}$$

$$E_{cpu-state} = \sum_{i=1}^{m} P_{cpu-state}(i) T_{cpu-state}(i) \quad \textbf{3.6}$$

$$E_{cpu-change} = \sum_{j=1}^{n} N_{cpu-change}(j) \left(T_{init-end}(j) \left(P_{init}(j) + \frac{P_{end}(j)}{2} \right) \right) \quad \textbf{3.7}$$

Where E_{cpu} is the total processing module energy consumption, which is composed of two parts as show below:

- $E_{cpu-state}$ Which is the state energy consumption, that is a function of the consumed power in that state $P_{cpu-state}$ and time the node stayed in it $T_{cpu-state}$, where m represents the number states the node processor went into.
- $E_{cpu-change}$ Which is the transition energy consumption, that is a function of the times the processor changed state $N_{cpu-change}$, and the time consumed in the transition $T_{init-end}$, the power consumed in the originating P_{init} and terminating P_{end} states, where n is the number of transitions.

In order to formulate some Figures out of the above equations, typical values for different parameters have to be adopted, the values provide in [32] are reasonable and convincing, they are listed in Table 3.4 below:

State	Power(mw)
Run	400
Idle	50
Sleep	0.16

State transition Class	State transition time (us)
Trun-idle	10
Tidle-run	10
Tidle-sleep	90
Trun-sleep	90
Tsleep-run	160

Table 3.4: Typical parameters for the processing module energy consumption [32]

If we plug the above numbers into the relative equations, the results come out as shown in Table 3.5 below, which shows how much energy would a transition from a certain state to another state would cost, the last column shows how much energy would be consume if we stay one second in a state after the transition.

Energy Model	E cpu-state init (W)	E cpu-state end (W)	State transition time (S)	E cpu-change (J)	E cpu (J) if the CPU stay in the state for one second
Run-idle	0.4	0.05	1×10^{-5}	225×10^{-8}	0.05
Idle-run	0.05	0.4	1×10^{-5}	225×10^{-8}	0.4
Idle-sleep	0.05	16×10^{-5}	9×10^{-5}	2.25×10^{-6}	16×10^{-5}
Run-sleep	0.4	16×10^{-5}	9×10^{-5}	1.8×10^{-6}	17×10^{-5}
Sleep-run	16×10^{-5}	0.4	0.16	32×10^{-3}	0.43

Table 3.5: Typical energy consumption Figures by a Processing module [32]

In order to illustrate more, Figure 3.18 below was produced from the Table 3.5 to compare and find out which stage uses the largest amount of energy, the result is clear, it shows that the run mode is using up to 90% percent of the energy while the idle mode comes in the second place with about 9.5% and the rest of state and transitions are negligible when compared with them.

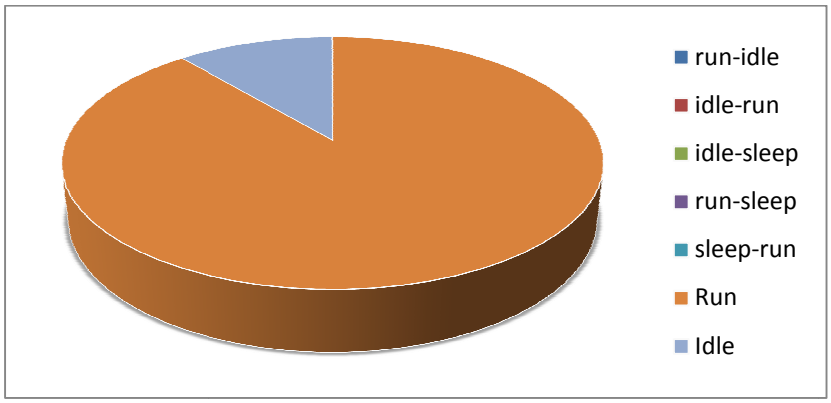


Figure 3.18: Energy conumption comparison in processor module [32]

3.6.3 Transceivers Subsystem Energy Consumption

The transceivers are the largest consumer of the nodes energy since they have to transmit the nodes sensored and processed data, in addition to relaying the other nodes data through it towards the actor and control interface direction.

The amount of energy a Chipcon CC2420 transceiver consumes, as in the case of other WSN node modules, depends highly on the role and position of the WSN node, it will use more energy if it's a hub of other nodes and less if it's only serving its own requirements.

The transceiver rolls around between six states of operation as shown in Figure 3.19 below, using nine possible transoms.

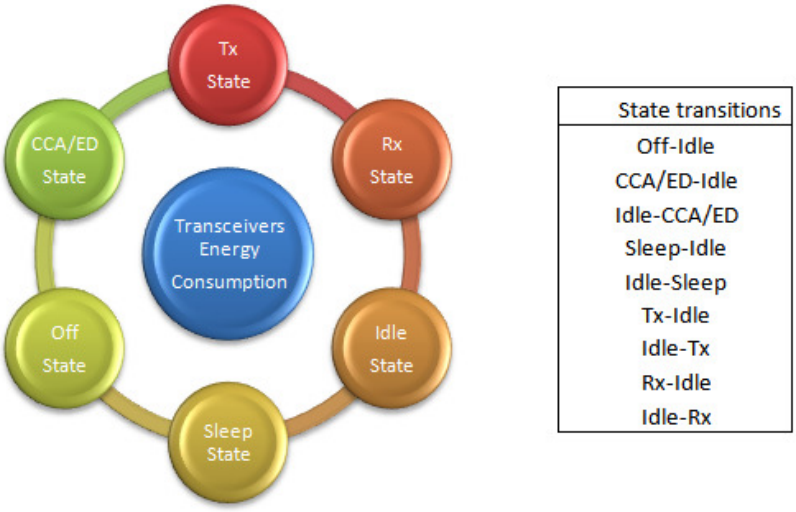


Figure 3.19: Transceivers States and Transitions of operations [32]

The main states that consume energy are the Tx and Rx, where the transceiver is transmitting or receiving data to or from other nodes, the energy consumed increases the longer the distance of communication and as the size of the communicated data becomes larger, and vice versa.

The other states consume less energy; those states are described below briefly:

- CCA/ED State (Clear Channel Assessment Energy Detection), the transceiver goes into this state befor transmitting to avoid excessive retransmission due to collisions.
- Idle State, the transceiver goes into this state once it finishes transmission, and there are no incoming messages from other nodes.
- Sleep State, in this state the transceiver Rx and Tx functionalities are off, but a small circuit monitors any activities and turns on the transceiver if needed.
- Off State, the transceiver is completely tuned off and can be turned on if there is something to transmit by the processing and sensing modules.

3.8 And 3.9 describe a method to calculate the energy consumed in the different modes of operation of the transceiver states shown previously:

$$E_{trans-state} = E_{TX} + E_{RX} + E_{Idle} + E_{sleep} + E_{CCA} \qquad \textbf{3.8}$$

$$E_{trans-state} = \sum_{i=1}^{N_{TX}} \frac{V_o I_{TX} L_i}{R} + \sum_{i=1}^{N_{RX}} \frac{V_o I_{RX} L_i}{R} + V_0\left(I_{Idle} T_{Idle} + I_{sleep} T_{sleep} + I_{CCA}\, T_{CCA}\right) \qquad \textbf{3.9}$$

Where E stands for the consumed energy in the different states of the transceiver, L is the length of a message the transceiver processes in a certain state, T is the time transceiver stays in a state . Which at the end is summarized by the voltage V and current I used in any state in order to compute the state energy consumption and R is hop length.

3.10 And 3.11 below describes the amount of energy consumed white the transceiver moves from one state to another:

$$e_{trans-change} = \frac{T_{init-end}(P_{init} + P_{end})}{2} \quad \textbf{3.10}$$

$$e_{trans-change} = \frac{V_0 T_{init-end}(I_{init} + I_{end})}{2} \quad \textbf{3.11}$$

Where e in the transition energy cost, and T is the time required to make the transition, while P_{init} and P_{end} are the required power of the initial and final state of the transition respectively which is finally expressed by the amount of voltage V and current I usage during the transition.

In order to formulate some Figures out of 3.10 and 3.11, typical values for different parameters have to be adopted, the values provide in [32] are reasonable and convincing; they are listed in the Table 3.6 below:

State	Current
V_0	5 v
Ioff	0.02 µa
Itx	17.4 ma
Irx	19.7 ma
Iidle	426 µa
Icca/ed	17.4 ma
Isleep	20 µa

State transition Class	State transition time
Toff-idle	1 ms
Tcca/ed-idle	2 µs
Tidle-cca/ed	192 µs
Tsleep-idle	0.6 ms
Tidle-sleep	192 µs
Ttx-idle	2 µs
Tidle-tx	192 µs
Trx-idle	2 µs
Tidle-rx	192 µs

Table 3.6: Typical parameters for the Transceiver module energy consumption [32]

The Table 3.7 below shows the energy consumption of a typical WSN node transceiver, clearly it shows that the Rx and Tx states are the major among all the states:

E trans - state	I (A)	V (v)	Power (w)	State Energy (J) if Node stays 1 second at the state
E off	2×10^{-8}	5	1×10^{-7}	1×10^{-7}
E tx	17×10^{-3}	5	87×10^{-3}	87×10^{-3}
E rx	19×10^{-3}	5	98×10^{-3}	98×10^{-3}
E Iidle	4×10^{-4}	5	2×10^{-3}	2×10^{-3}
E Icca/ed	17×10^{-3}	5	87×10^{-3}	87×10^{-3}
E Isleep	2×10^{-5}	5	1×10^{-4}	1×10^{-4}

Table 3.7: Typical energy consumption Figures by a Transceiver module states [32]

The Figure 3.20 below highlights the most demanding states in terms of energy consumption in the transceiver module:

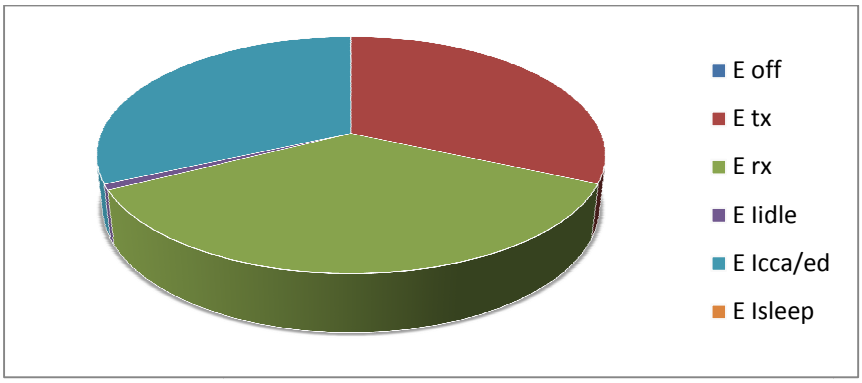


Figure 3.20: Tranciever states energy usage destribution [32]

The Table 3.8 below shows the energy cost of a transition in a typical transceiver, the most consuming transition is when the transceiver moves from idle state to receiving state.

E trans - change	State transition time	P init state (w)	P end State (w)	E trans - change (J)
Toff-idle	1×10^{-3}	1×10^{-7}	2×10^{-3}	1×10^{-6}
Tcca/ed-idle	2×10^{-6}	87×10^{-3}	2×10^{-3}	89×10^{-9}
Tidle-cca/ed	19×10^{-5}	2×10^{-3}	87×10^{-3}	85×10^{-7}
Tsleep-idle	6×10^{-4}	1×10^{-4}	2×10^{-3}	66×10^{-8}
Tidle-sleep	19×10^{-5}	2×10^{-3}	1×10^{-4}	21×10^{-8}
Ttx-idle	2×10^{-6}	87×10^{-3}	2×10^{-3}	89×10^{-9}
Tidle-tx	19×10^{-5}	2×10^{-3}	87×10^{-3}	85×10^{-7}
Trx-idle	2×10^{-6}	98×10^{-3}	2×10^{-3}	1×10^{-7}
Tidle-rx	19×10^{-5}	2×10^{-3}	98×10^{-3}	96×10^{-7}

Table 3.8: Typical energy consumption Figures by a Transceiver module state transitions [32]

It is very clear from the Figure 3.21 below that the transition from the idle mode to the transmitting and receiving states is consumes a large amount of energy compared to other transitions.

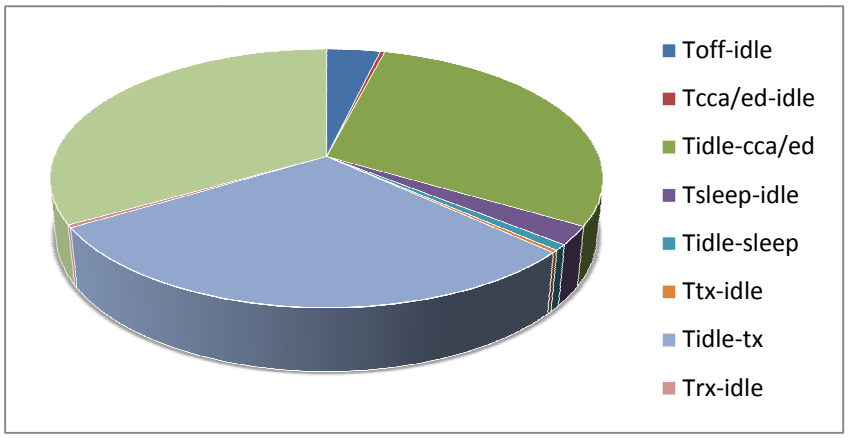


Figure 3.21: Tranciever transitions energy usage destribution [32]

3.6.4 Adopted Refined Energy Model

To evaluate the energy consumption efficiency of DEMA, a customized energy model should be developed since implementing all the above strictly would require very detailed level simulation and implementing a communication messaging that is out of the scope of this work.

As for the processing module, the energy used will be the same in our algorithms, or in any other algorithms since it is not in the performance indicators of node placement algorithms, as for the sensing module of the sensor node the previous model is fully implemented, we consider the cost of one sensing period in terms of energy:

E_{sensor}	=	4×10^{-3}	Joules

Regarding the transceiver energy consumption, it goes through many stages while its working from sleep to idle, then transmit and receive, and checking the radio environment all the time, only the transmit and receive energy costs are considered since all the other states are relatively small, and cancel out when comparing the performance of the different protocols since they are fixed, A typical message would take 6.4 us, below are the Figures for energy consumption for receiving and sending a typical message :

E_{tx}	=	0.56	ujoules at 200m
E_{rx}	=	0.64	ujoules at 200m

3.12 below models the network energy consumption, here n is the number of hops from the node to actor with CTR =100 m:

$$E_{Consumed} = n\left(\frac{(E_{tx} + E_{rx})}{2} + E_{Sensor}\right) \qquad \textbf{3.12}$$

3.7 Topology Maintenance

Topology maintenance is a new add on design approach to wireless sensor network, to increase the network reliability and topology resiliency to failures.

In general when a WSN is deployed, the previously described algorithm is initiated, when it's finished, the resulting network will be arranged in clusters of sensors with one actor in each located in a position that will satisfy the minimum latency and energy consumption by the whole network, this phase is called topology construction phase.

The above topology is valid until sometime, where some node die out either because of low battery or could be damaged, at that case, the current topology loses its advantages in terms of the required optimization objectives, so a new phase is triggered where DEMA is to be repeated in order to optimize the network topology to maintain the same required level of efficiency, this is called Topology maintenance phase.

The maintenance phase where network recalculates the topology can be triggered based on different criteria such as in [9]:

- Time based triggering, where the maintenance happens periodically.
- Failure event triggering, to react to node failures.
- Energy based, to achieve fairness in node energy depletion.
- Density based triggering, to make the topology follow a uniform distribution.

In our work, the time based and failure triggering are combined together to initiate a maintenance process for the network topology, because they are passive, in fact they don't need any new messaging overheads, while the other methods are highly complicated and require extra communication between the nodes.

The Figure 3.22 below shows how this adds on is integrated into the algorithm:

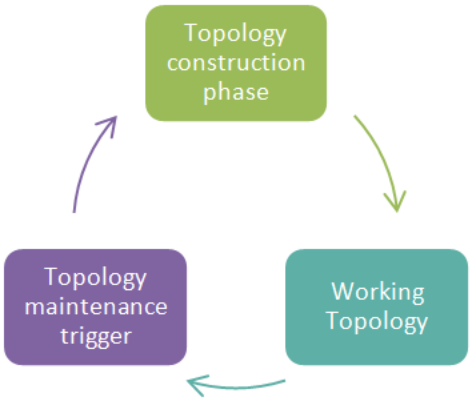


Figure 3.22: Network topology maintenance add on

The previously outlined algorithm achieves the preset network performance targets, with low complexity and in a distributed manner, where no need for extra communication messages between the actors, but still the algorithm is conflict free.

The Figure 3.23 is a detailed complete pseudo codes implementation of the algorithm, the MATLAB implementation program that reflects the procedures below is supplied in the appendices at the end of the thesis.

```
% As an input to the algorithm the number of actors N and the
% area boundaries Area are given.
Subarea[N,Area];
If N = 1
Actorlocation = Center(Area)
Else
Find(Area,area1,area2)
Area1= Subarea[(N/2),Area]
Area2= Subarea[(N/2),Area]
 End
    % The output of the above step is a set of subareas and
    % corresponding centers, next each actor calculates the
    % closest subarea and moves to it.
    For i := 1 to N
    For j := 1 to N
    Actortocenter[i,j] =sqrt ((X_A − X_C)^2 + (Y_A − Y_C)^2)
    End
    We take the minimum distance between any actor and the
    Subarea centers sand move the actor to it.
    For i := 1 to N
    Locationofactor [i] = Center with Min(actortocenter[i,:])
    End
    Final step is to move the Actor to the selected center
    Move Actor[i]to locationofactor [i]
    End
    % As an input to this stage an n x 3 vector is provided
    % where its entries are as explained before are a triplet
    % as follows (Node id,X location, Y location),and n is the
  % number of nodes, CTR is the critical transmissionrange.
    Nodelocation[n,3] = (Noden,Xn,Yn);
    Distacemtrix[i,j];
    For i := 1 to n
    For j := 1 to n
    Distacemtrix[i,j] =sqrt ((X_I − X_J)^2 + (Y_I − Y_J)^2)
    Adjacency[i,j];
    For i := 1 to n
    For j := 1 to n
                          { 1, DistaceMtrix[I,j] < CTR
    Adjacency[i,j] =      { 0, DistaceMtrix[I,j] ≥ CTR
    % The output of the algorithm is an n x n matrix where its
    % entries path(i,j)= the number of hops between nodes i & j.
    % As an input to the algorithm an n x n adjacency matrix is
    % provided, where its entries are 1 if there is an
    % edge(i.j)and 0 otherwise
    Adjacency[i,j];
    Floydwarshall (Adjacency)
    For k := 1 to n
    For i := 1 to n
    For j := 1 to n
    Path[i, j] = min (Adjacency [i,j], Adjacency [i,k]+ Adjacency [k,j] );
```

```
49.     % The output of the algorithm is an n x n matrix where its
50.     % entries path(i,j)= the number of hops between nodes i & j.
51.     % As an input to the algorithm an n x n all pair shortest
52.     % path matrix. Is provided.
53.     Path[i, j];
54.     For i := 1 to n
55.     Avg{i}= Average path[i, :];
56.     Number of Ones[i] = Ones path[i, :];
57.     Number of Tows[i]= Tows path[i, :];
58.     End
59.     Avg Location = the sensor location with Min (Avg{i});
60.     Ones Location=sensor location with Highest (Number of Ones[i] {i});
61.     Twos Location= sensor location with Min (Number of Tows{i});
62.     New Actor Location = (Avg Location + Ones Location + Twos Location )3;
63.     Move the actor to the new location;
64.     If (5 % of sensors die Or T0 elapses)
65.     Restart algorithm.
```

Figure 3.23: DEMA pseudo code

4.DEMA Simulation Framework and Results

This chapter presents and discusses the results of DEMA.

- ☐ Benchmarking algorithms (Random, Uniform, and COLA)
- ☐ Simulated setup
- ☐ Simulation Performance
- ☐ Simulation results

In this chapter the final results proving the superiority of DEMA are laid out comparing the performance of DEMA with three other algorithms:

- Random deployment algorithm, which represents how things will be if the positions of the actors are not carefully planned.
- Uniform deployment algorithm, which brings a little order into the actors positions which enhances the network performance.
- COLA algorithm, which is a very well optimized algorithm in terms of actor position selection.

In the following sections, a brief explanation of the above algorithms, alongside the inputs and simulation procedure is provided.

4.1 Random Algorithm

Random algorithm is simply as the name implicates, randomizes the positions of the actors and sensor nodes at the initial deployment phase only and does not move thereafter, below some of the characteristics of the algorithm:

- Node positions are static.
- Energy consumption model is using full node power to communicate.
- It is impractical to apply clustering due to randomness.

Basically, random algorithm represents the status of the WSN network when it is rolled out.

4.2 Uniform Algorithm

Uniform algorithm, brings some order to the chaos of the initial network deployment, by placing the actors in a calculated position that guarantees that each actor node would be at the centriod of a subareas as described below:

- Deployment area is equally divided.
- Actors are placed at the subarea centers.
- No clusters are formed on sensor actor level.
- Energy consumption is modeled the same as in the random algorithm (Fixed power).

In this algorithm the placement of the actors is well studied, and results in an improvement in the network performance when compared to random algorithm performance.

4.3 COLA Algorithm

COnnectivity and Latency aware Actor node placement algorithm, enhances the results of the uniform algorithm, it goes into five steps to achieve better delay and actor network coverage:

- Network bootstrapping where the nodes exchange their locations.
- Actor nodes compute and divide the deployment area into clusters and smaller areas, where each actor computes the nearest area center and moves towards it.
- The entire sensor nodes within each cluster associate with the actor at its center.
- On a cluster level, the cost of communication between all nodes is computed using Floyd-Warshall algorithm.

- The actor is moved to the location of the sensor node that has least communication delay with other nodes in its cluster.

COLA was discussed thoroughly in [5], below is a brief listing of its main characteristics:

- Only sensor nodes location is considered.
- Energy consumption is based on full power communication mode.

COLA proved to be a greate enhancement in placing actor nodes when compared to the previously described algorithms, Figure 4.1 below shows the pseudo code of COLA.

```
1  The actor node computes the minimum distance matrix for
   the cluster:
        M=d[i, j]  for all i, j ∈ Sensors in the cluster
2  FOR each row k of M
        /* Find the longest path for this node to all others */
3       Find Max(row(k));
4       MaxList ← Max(row(k));
   END FOR
   /* Pick shortest entry in the set of longest paths*/
5  MinCost ← Min(MaxList);
6  VertexCenter ← Pos(First node on Path(MinCost));
7  Relocate the actor to VertexCenter
```

Figure 4.1: COLA psedou code[5]

4.4 Simulation Setup

4.4.1 Targeted Application

The sensor nodes modeled using the energy consumption model that was outlined in the previous chapter are structured physically as shown in Figure 4.2 below, where the characteristics of hardware units that are related to our work were discussed in chapter 3:

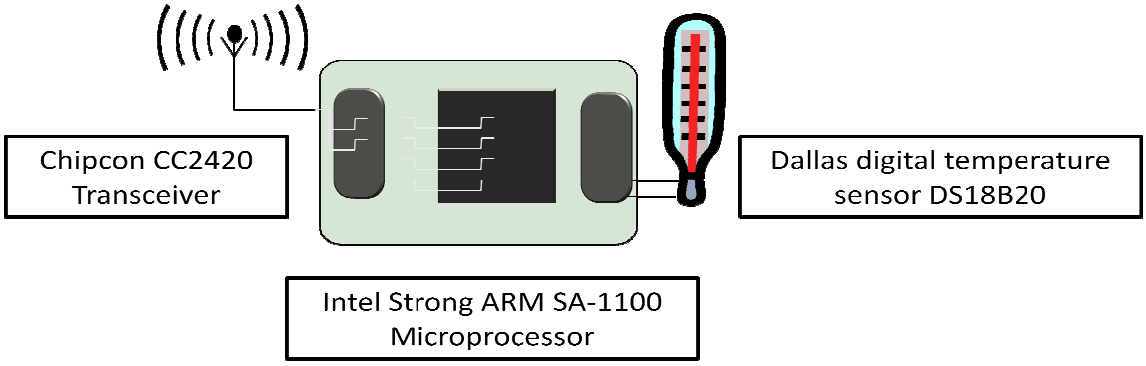


Figure 4.2: Physical structure of the sensor nodes

As a typical application for COLA and DEMA, object or event detection such as detecting fires in a forest and relaying the detected event to the actor to take an action to put down the fire or to raise an alarm. The temperature sensor can be exchanged with motion, acoustic or pressure sensors that can have very similar energy consumption model to be used for other applications.

4.4.2 Simulation Inputs

The simulation of the previous algorithms, in addition to the developed one was implemented in MATLAB; the simulation required the following inputs:

- Deployment area boundaries, which was considered to be a squared flat area of 2000 meters length edges.
- The number of sensor nodes in the deployment area, which was fixed to 600.
- The transmission range of the nodes which was considered to be 100 meters.
- The number of available actors was varied in steps of 3 between 1-30 actors.

The above parameters were chosen according to the typical accepted numbers in the literature [3, 5, and 6], for example 100 meters transmission range is very well established in the literature, as for the number of sensors, it's typical to achieve a fair node density within the deployment area.

4.4.3 Simulation Procedure

In order to achieve the study goals on network performance optimization, in terms of energy consumption and communication latency, the following procedure has been adopted:

- Random, Uniform, COLA, and DEMA were run using the same inputs above.
- The number of actors was changed in steps of three starting from one actor to 30 actors.

A sample run of the procedure is shown in appendix A, showing the different outputs of each algorithm stages.

4.4.4 Simulation Metrics

The most important metrics that were taken into consideration while the simulation was performed are the following:

- Energy
- Delay

4.5 Statistical Validity of the Results

In order to measure the statistical accuracy and validity of the obtained results the, a confidence interval of 95 % was established.

The average standard deviation of the sample runs for the DEMA and COLA algorithms for both the energy and the delay is 0.6 (u Joules and Hops respectively) , In order to obtain the 95 % confidence interval for the results at 70 degrees of freedom (number of simulation runs) the corresponding t student value is 0.1994.

Applying the above values to the t student test the following inequality indicates that the calculated results are within an error margin of 0.142 from the true value:

$$\mu_{Runs} - 0.142 \leq \mu_{True} \leq \mu_{Runs} + 0.142$$

As an additional indicator to the validity of the simulation, the results of each run for DEMA stay within 10 % of the average values for COLA and DEMA results.

4.6 Delay Optimization

Table 4.1 below shows the average delay from any node in the WSN network will experience when communicates with the actors, it can be seen clearly that the rando algorithm is the worst while DEMA is the best

Average Delay (Hops)				
Actors	Random	Uniform	COLA	DEMA
3	18.3	17.8	11.6	11.2
6	18.8	17.9	9.5	7.5
9	19.4	17.4	7.3	6.1
12	19.5	17.9	6.3	5.6
15	19.2	18.7	5.5	4.9
18	19.9	18.9	5.1	4.5
21	19.1	19.1	4.7	4.2
24	18.9	19.1	4.4	4.0
27	19.2	18.9	4.1	3.7
30	19.5	18.2	3.8	3.5

Table 4.1: Average delay results for different algorithms

Figure 4.2 below highlights the fact that the uniform algorithm is better in terms of delay than the random one, but still both of them fail to archive DEMA results.

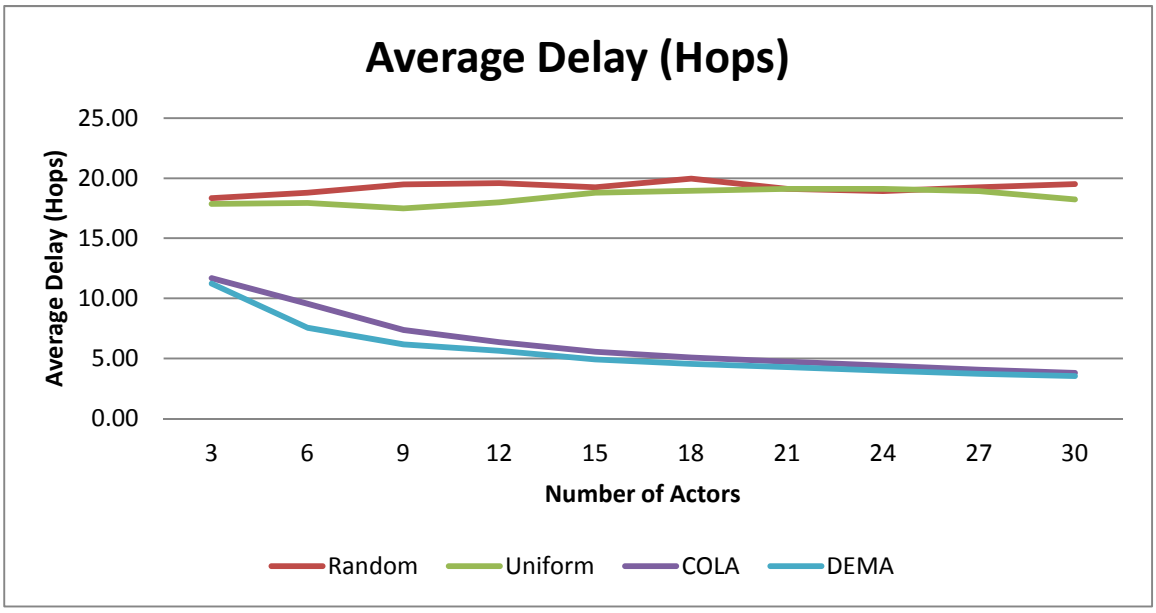


Figure 4.3: Average Delay comparison between all algorithms

Figure 4.3 is a one to one comparison between the COLA algorithm and the developed one, which shows that we achieved better results with our algorithm in minimizing the average delay, especially at point with lower actor densities in the network, but still DEMA maintains its superior performance at higher number of actors in the network.

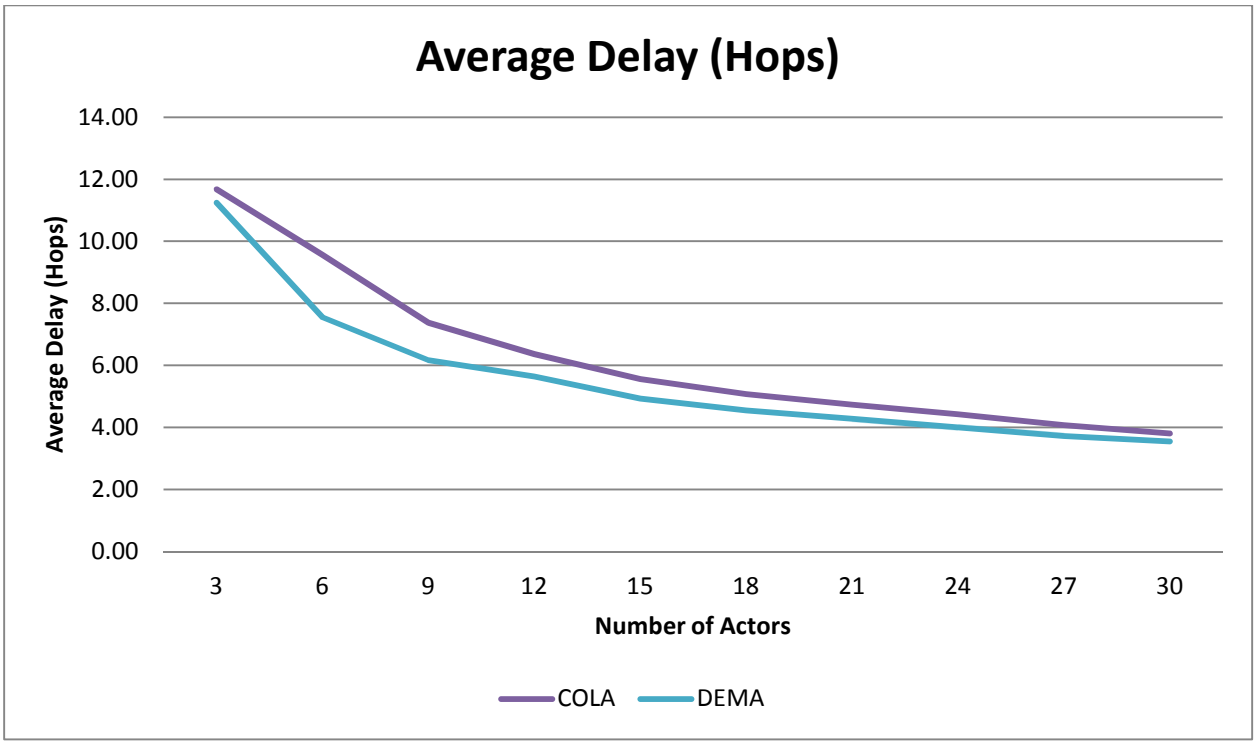


Figure 4.4: Average Delay comparison COLA vs. DEMA

4.7 Average Energy Saving

Table 4.2 below sums up the results of the simulations measuring the average energy consumption of the WSN network, which again stresses the outstanding performance of DEMA, even in better results in the energy consumption.

Average Energy (u Joules)				
Actors	Random	Uniform	COLA	DEMA
3	22.0	21.5	14.0	5.9
6	22.6	22.5	11.5	5.1
9	23.4	23.4	8.9	4.9
12	23.5	22.9	7.6	4.3
15	23.1	22.9	6.7	4.5
18	24.0	22.9	6.1	4.4
21	22.9	21.8	5.7	4.4
24	22.7	22.4	5.3	4.4
27	23.1	21.7	4.9	4.4
30	23.4	19.2	4.6	4.4

Table 4.2: Average energy results for different algorithms

Figure 4.4 below shows performance of random, uniform, and DEMA, in the area of energy consumption

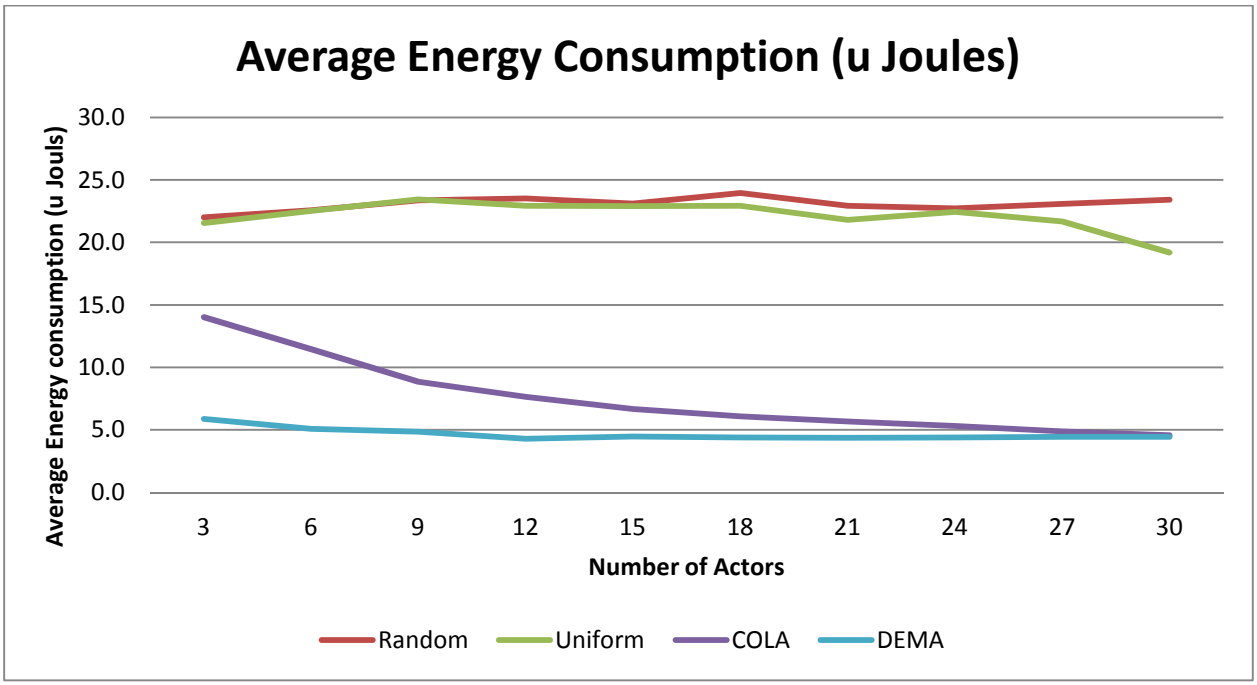


Figure 4.5: Average energy consumption between all algorithms

DEMA has not only proved to be the most efficient in energy consumption, but also the only scalable algorithm of all studied algorithms, as shown in the two Figures 4.4 and 4.5 where different energy model was used in evaluating DEMA performance as explained in chapter 3.

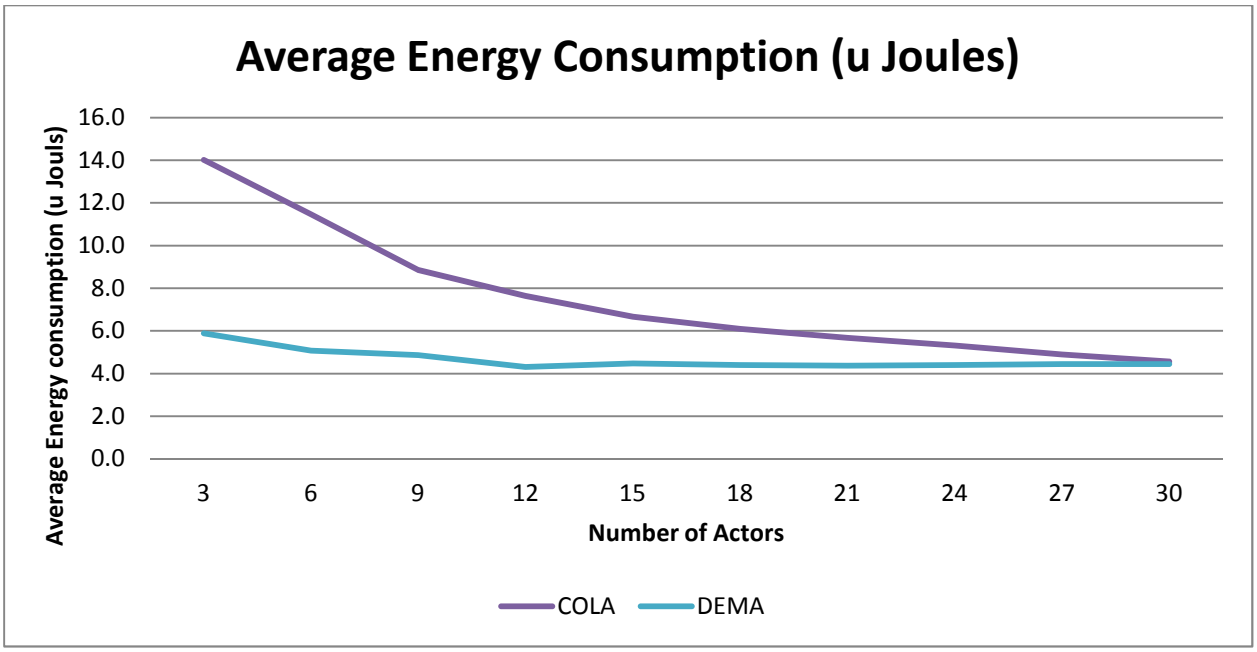


Figure 4.6: Average energy consumption comparison COLA vs. DEMA

4.8 Energy Saving with New Energy Model for COLA

A different set of simulations were conducted to evaluate the performance of DEMA against COLA when both are using the energy consumption model described in chapter 3. The results assure the better performance of DEMA in reducing the amount of consumed energy as shown in Figure 4.7.

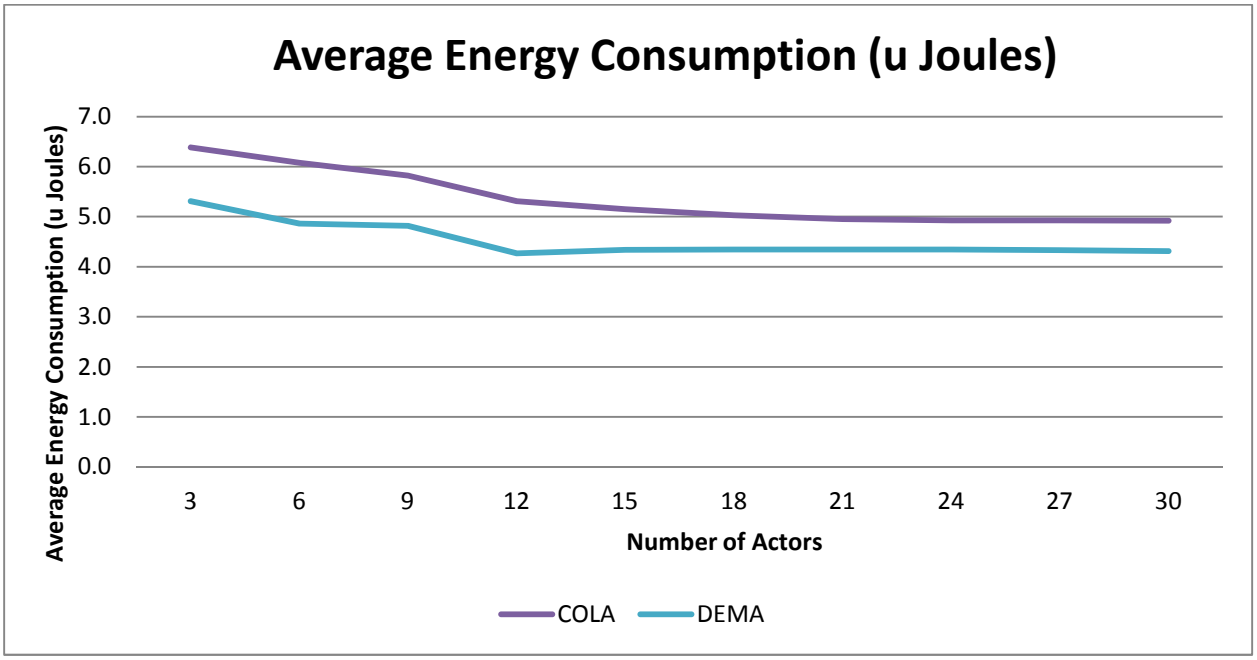


Figure 4.7 : Energy saving of DEMA compared to COLA when using the same energy consumption model

4.9 Overall Performance Results

In this section complete comparison charts are provided for each optimization objective to stress the novelty of our work. Figure 4.6 below shows the delay results for the studied algorithm while the ratio of the deployed sensors to actors is varied, clearly DEMA in this work is better than all of the

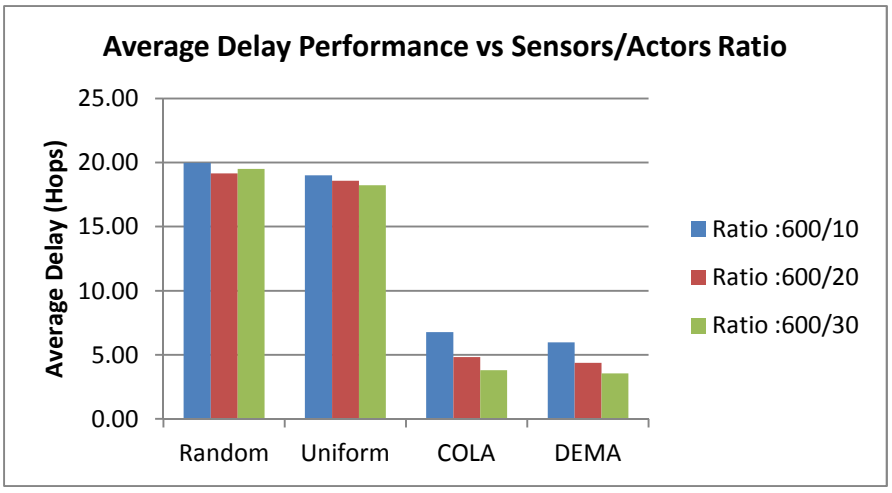


Figure 4.8: DEMA has the best delay performance

Figure 4.7 below shows the novelty of energy consumption of DEMA since it reduces the required energy with small number of actuators, which is clear when varying the ration of actors in the network..

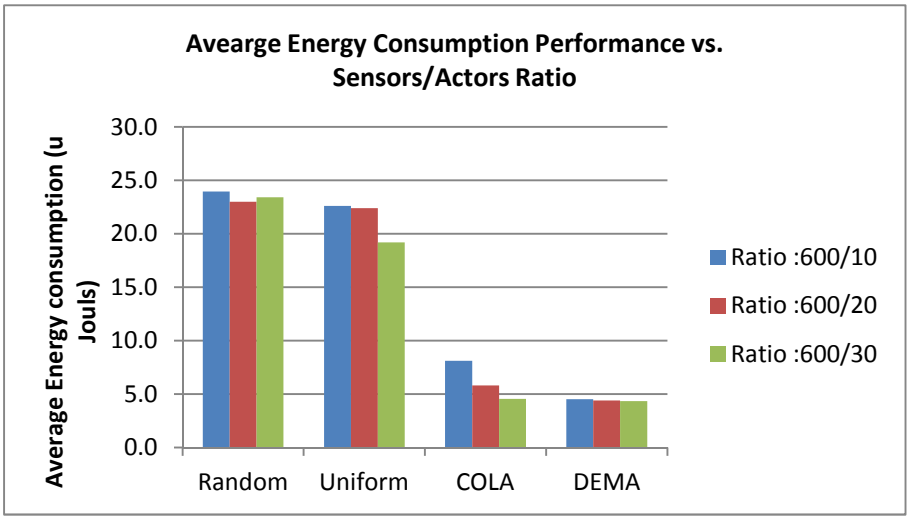


Figure 4.9: DEMA is the best energy performance

5. Conclusions and Future Work

This chapter concludes the work done in this thesis focusing on the obtained results.

- ☐ Key findings and contribution
- ☐ future works

This thesis addressed the node placement problem in wireless sensor networks, solving this issue would result in a WSN network that will be more efficient, provide better coverage and serve its purpose for a longer period of time with better performance than working without node positioning optimization.

The actor nodes placement strategy was considered in this work and not the whole sensor nodes in the network, since actors work as a hub and a sink between the sensor nodes and the command center

The study focused on finding locations to position actor nodes in a way that will minimize the average number of hops a sensor needs to go through to reach out to the nearest actor which in turn minimized the delay in the network. Also the same position is supposed to minimize the average energy consumed by the network.

An algorithm was developed to tackle the above problem by solving the all shortest path problem using the Floyd-Warshall algorithm after form clusters around the actors.

5.1 Key Findings and Contribution

DEMA achieved superior results when compared with general algorithms such as the random and uniform algorithm, and also when compared to algorithms developed within the same area and optimization mindset such as COLA [5].

The distinguishing points of the developed work can be summarized in the following two key findings on the network level:

- Minimum Average delay, DEMA achieved the best latency performance of all studied algorithms by at least 10%.
- Minimum average energy consumption, the performance of the proposed algorithm was superior in this category by a margin of at least 20%-30%.

Our work has introduced three more contributions in addition to the development of the new algorithm, as stated below:

- The addition of a topology maintenance phase to the algorithm that enables the network to readjust its status and stay optimized even though some of the sensor nodes fail down.

- The use of a precise energy measures supplied from practical studies, and adopting an energy consumption model that is more realistic in regenerating the results of related algorithms.

- Regenerating the COLA algorithm, with a higher studied number of nodes which resulted in exploring the algorithm performance beyond the range of the algorithm designers, with a better statistical resolution.

5.2 Future Work

The field of WSN node placement is vast and still relatively new, which makes it a good candidate to stay in the list of hot research topics in WSN and Ad-Hoc networks.

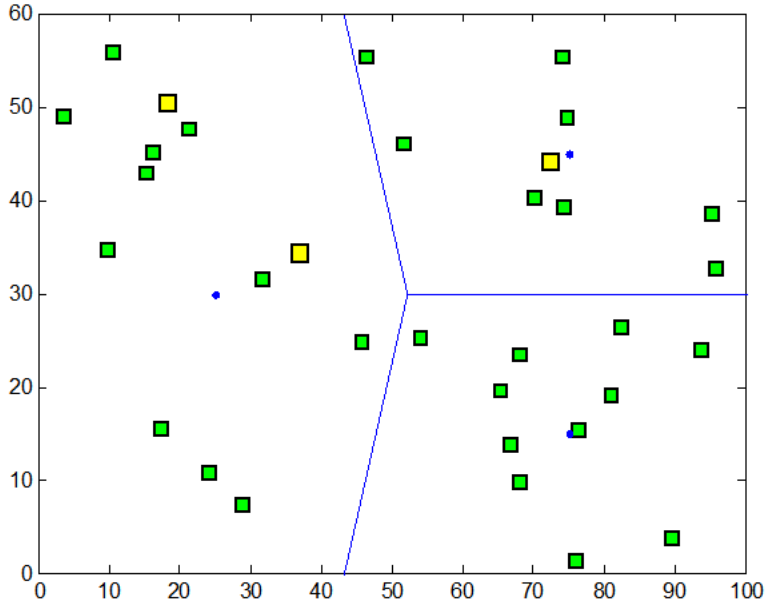
As always, it is not possible to design a complete solution for a problem that is still evolving and under development, because of that there is always a need for research optimization of previous work, as future additions that can be added to this works, we suggest the following:

- Extend the designed algorithms to whole sensor nodes and evaluating the cost of such an extension against the gained benefits if any.
- Benchmark the algorithm with new and other algorithms in the area.
- Extending the algorithm to WSN networks with more layers that work in a 3 D environment rather the 2 D environment we focused on.
- Test the results of the thesis against a practical results obtained from a real WSN physical test bid.

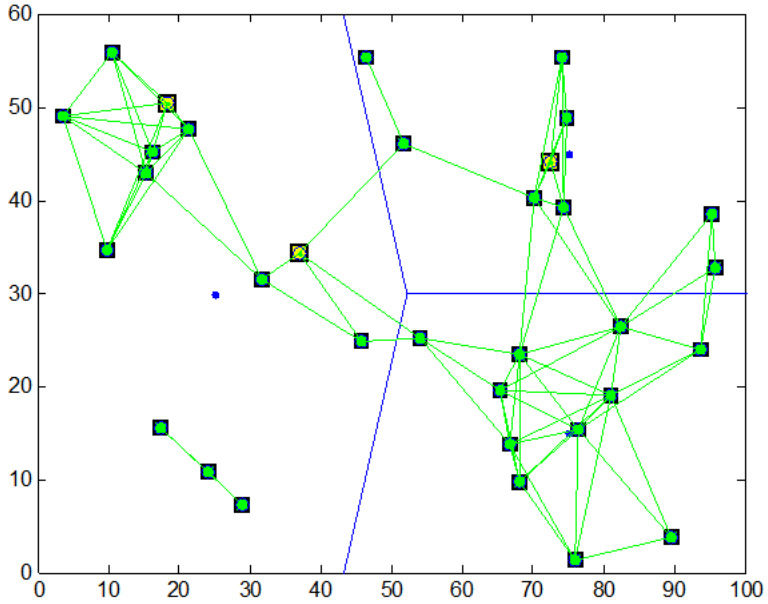
We hope that this work will be a seed for further research and work in this fascinating field of technology.

Appendix A : Sample Run with Results

In this appendix sample run of the simulation procedure is provided to show the way the results has been obtained, the Figure below shows a WSN network with 30 sensor nodes (Green) and 3 actors (yellow)



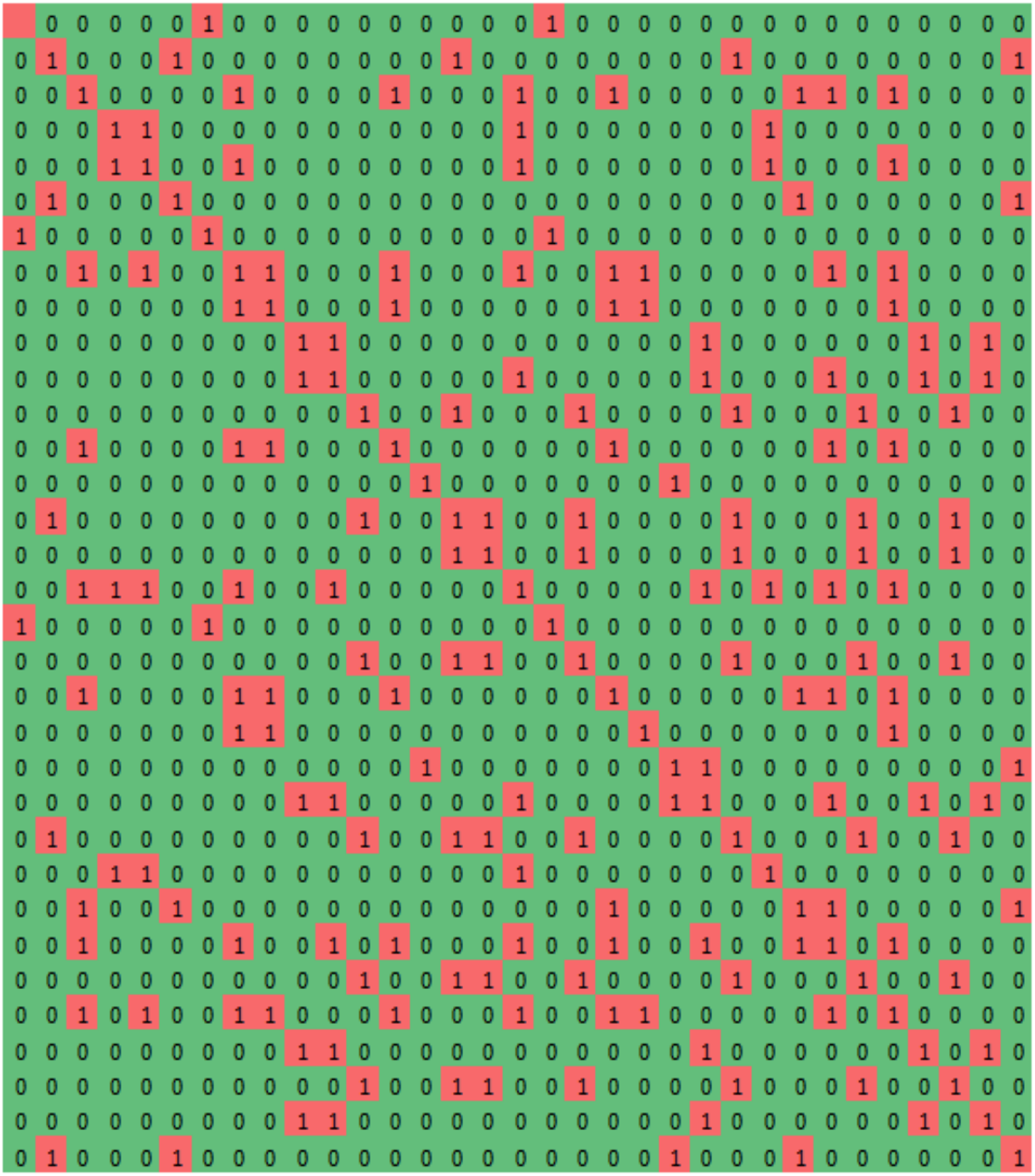
Below is a connected network of the above nodes:



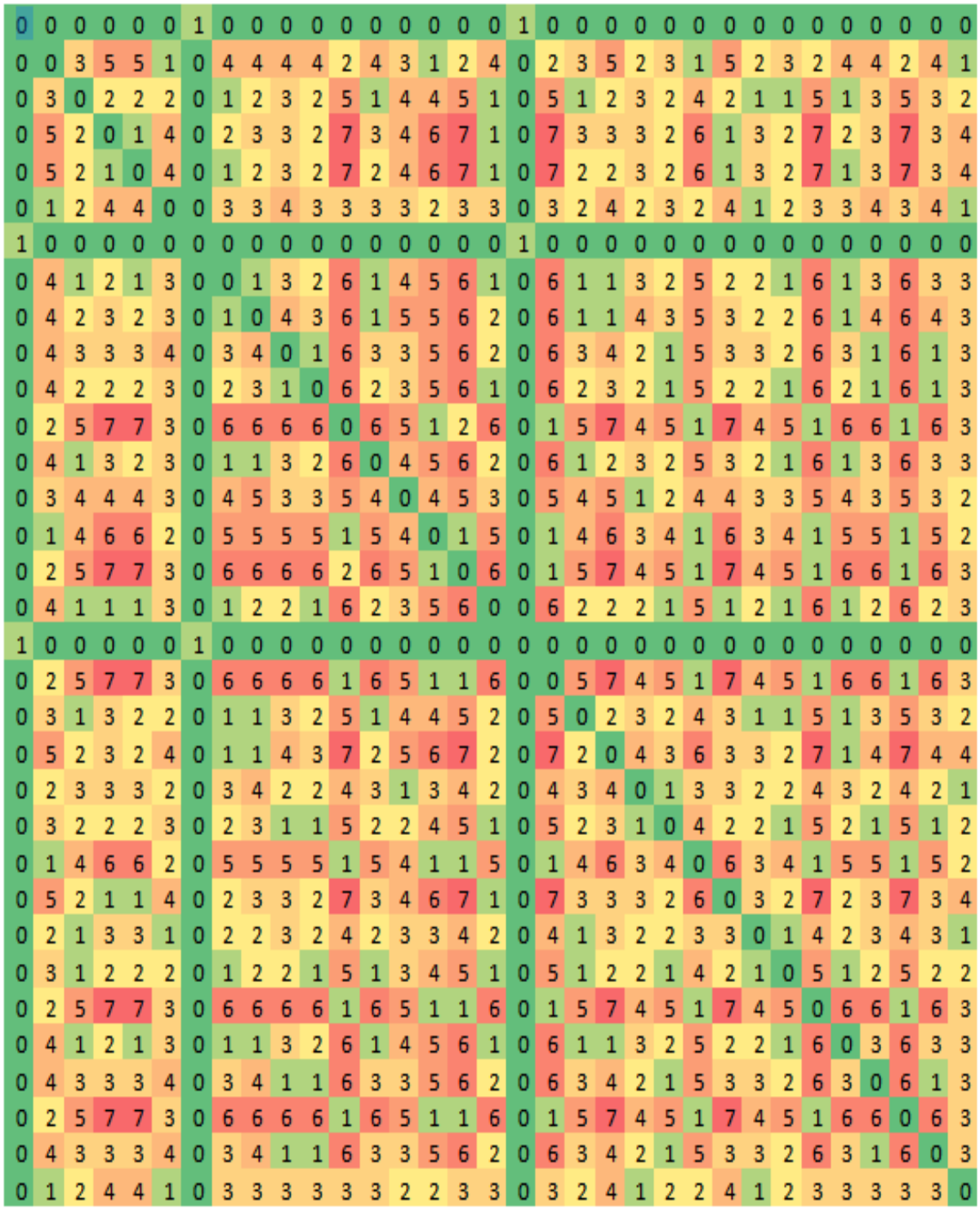
A.1 Random

Below the adjacency matrix and delay matrix are provided for the random algorithm

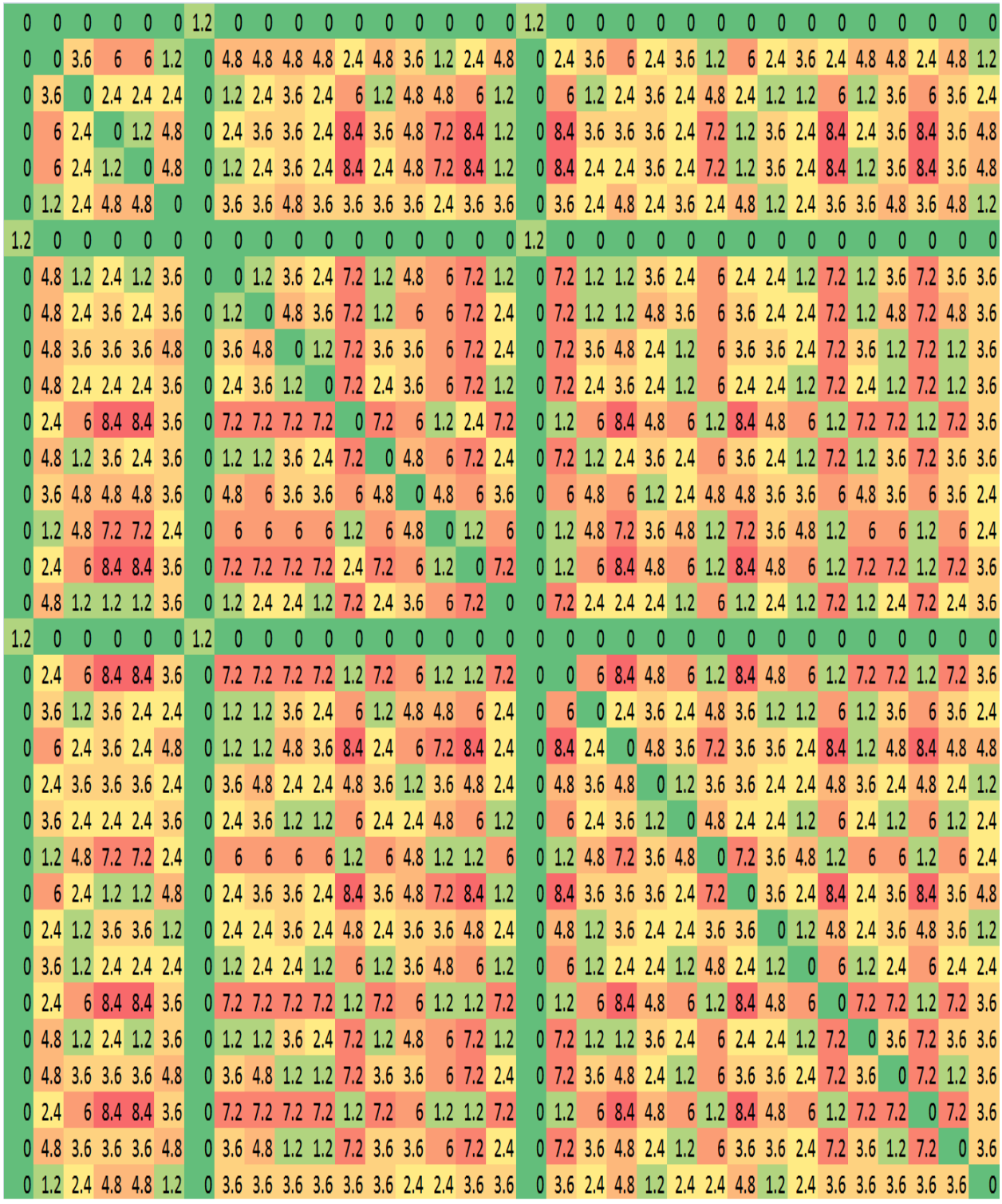
A.1.1 Adjacency Matrix



A.1.2 Delay Matrix



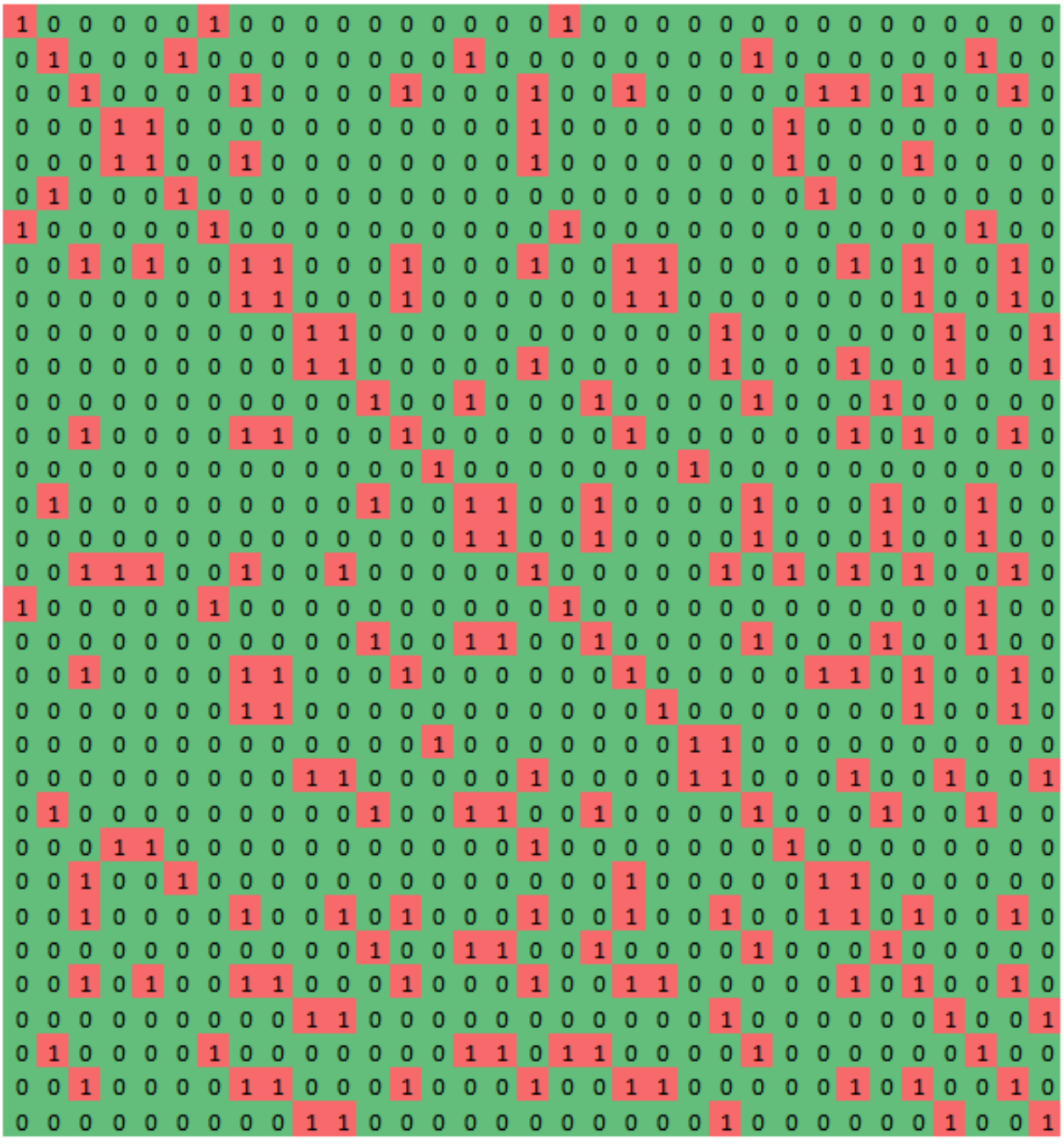
A.1.3 Energy Matrix



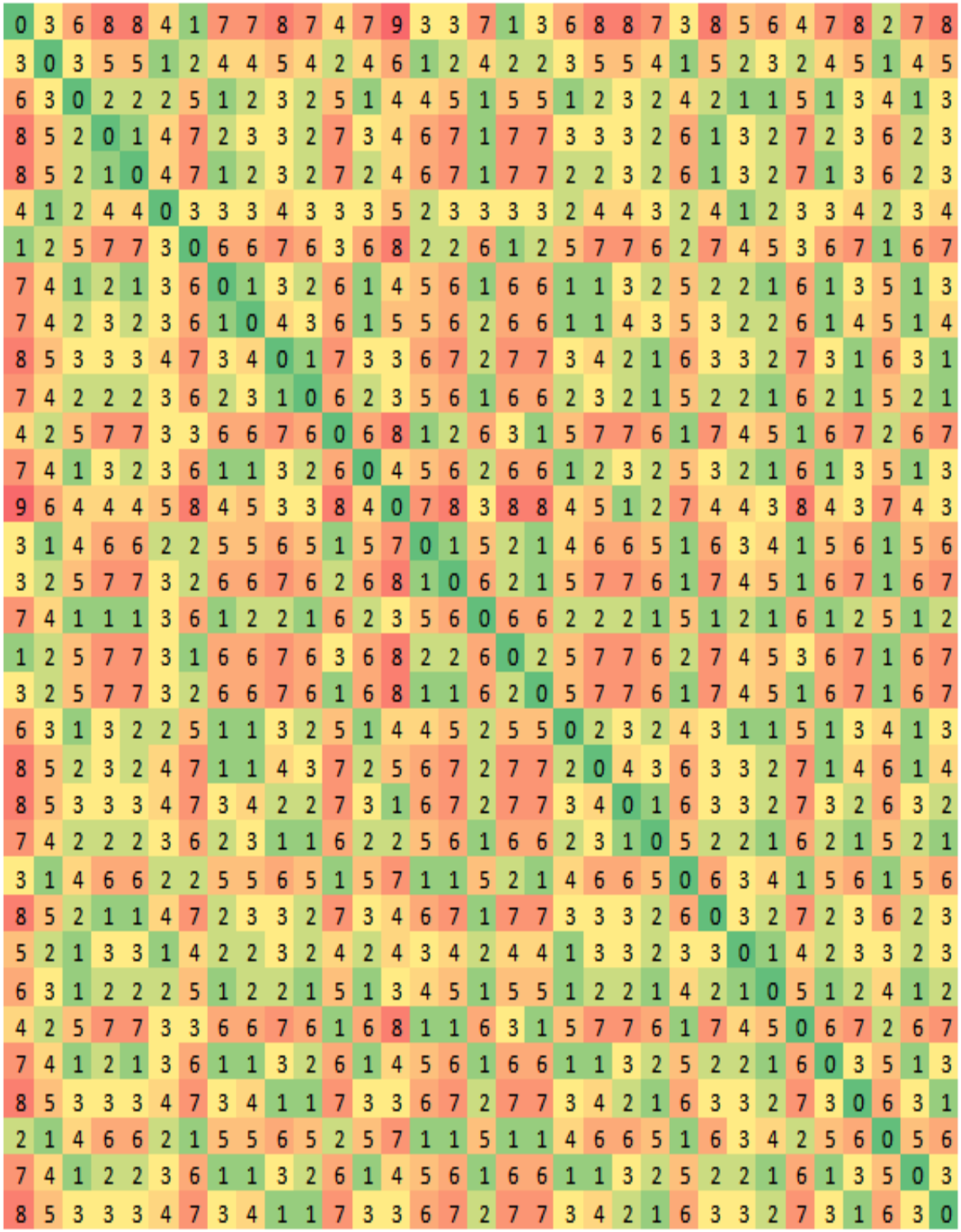
A.2 Uniform

Below the adjacency matrix and delay matrix are provided for the uniform algorithm

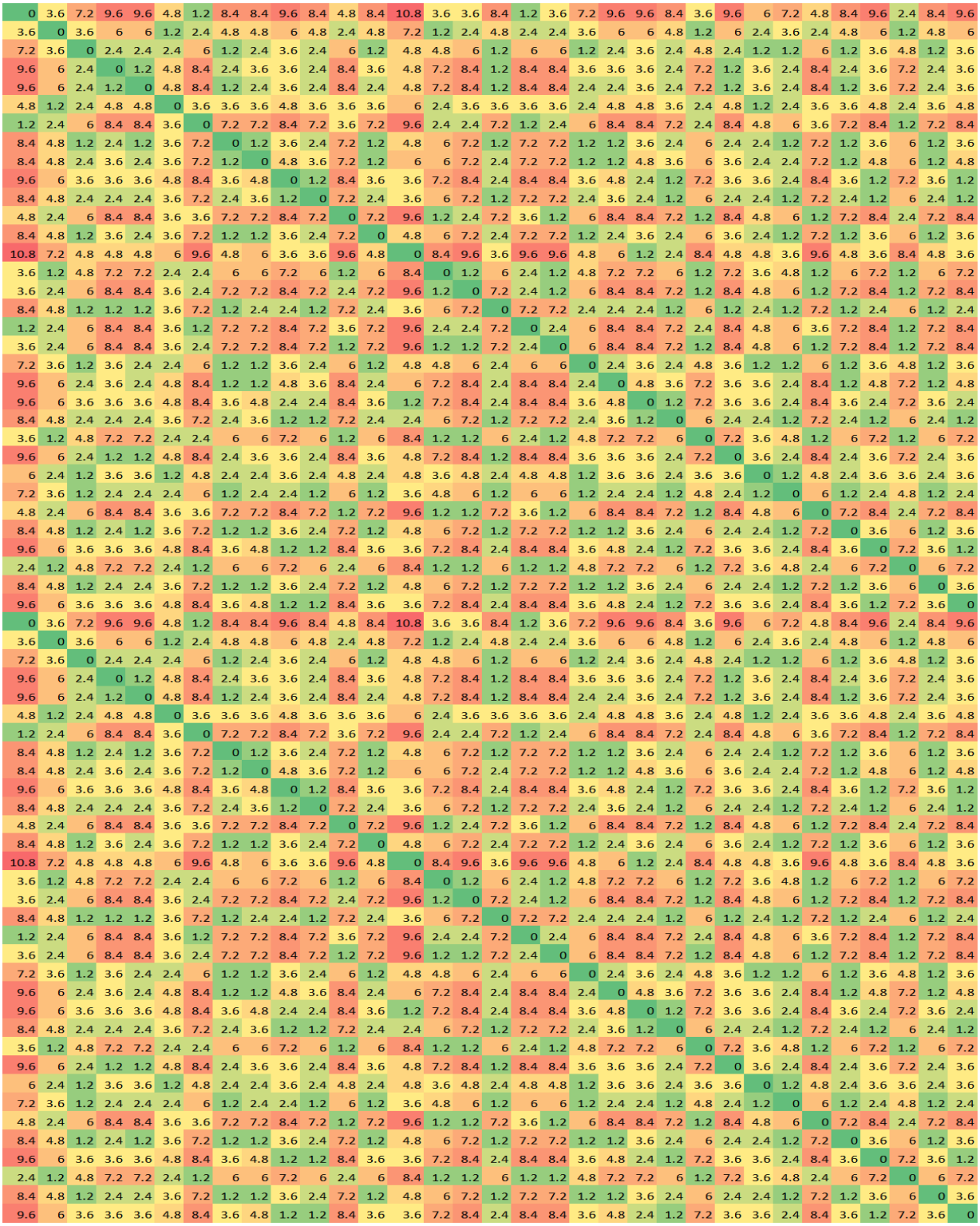
A.2.1 Adjacency Matrix



A.2.2 Delay Matrix



A.2.3 Energy Matrix

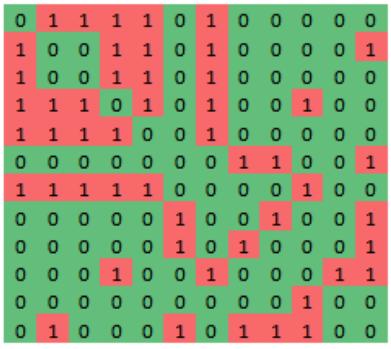


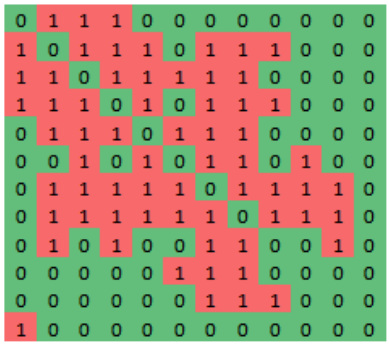
A.3 COLA

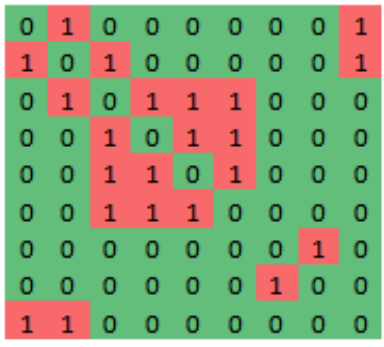
Below the adjacency matrix and delay matrix are provided for the COLA algorithm

A.3.1 Adjacency Matrix

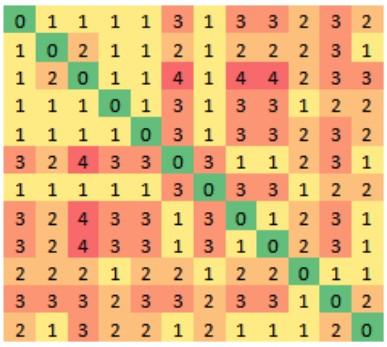
Here three adjacency matrices are available,since three clusters were formed:

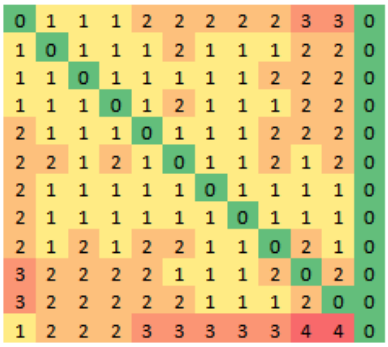


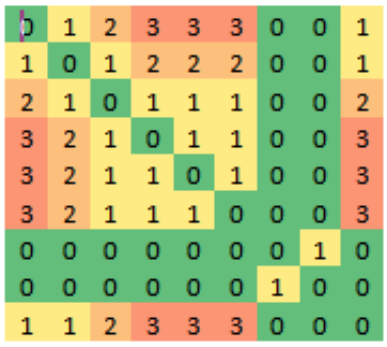




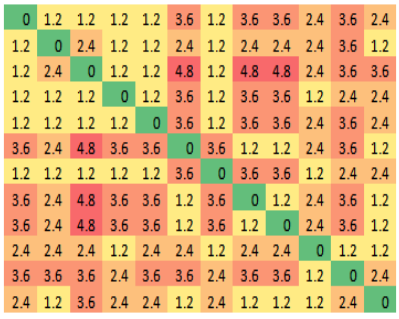
A.3.2 Delay Matrix

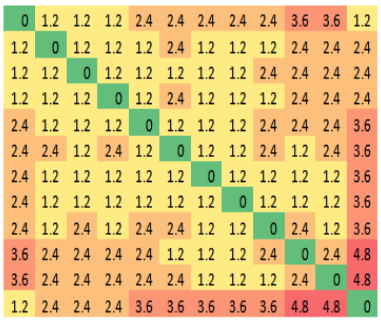


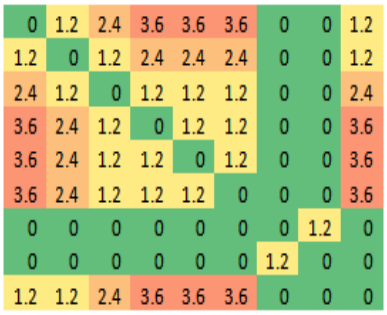




A.3.3 Energy Matrix





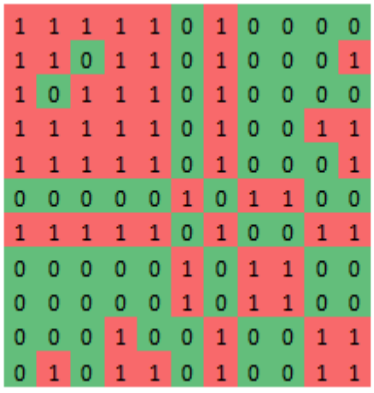


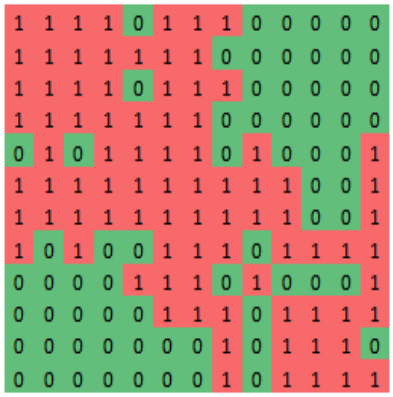
A.4 DEMA

Below the adjacency matrix and delay matrix are provided for DEMA.

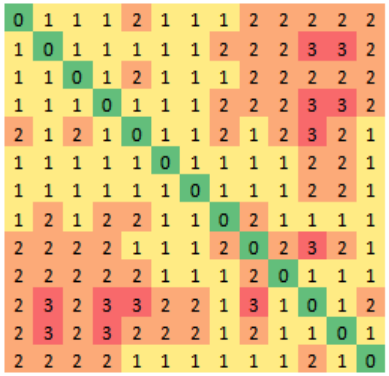
A.4.1 Adjacency Matrix

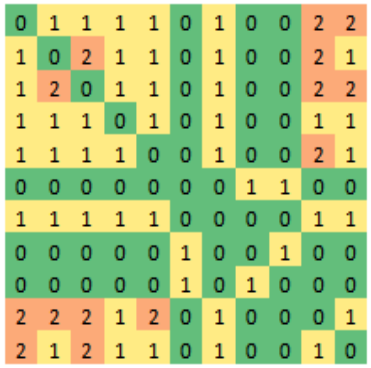
Here three adjacency matrices are available, since three clusters were formed:

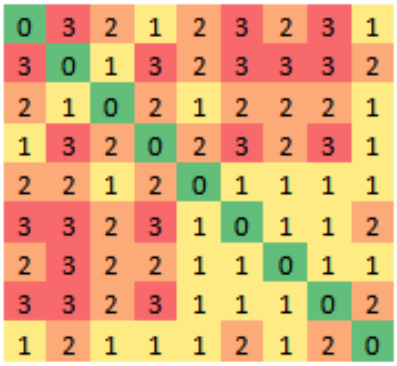




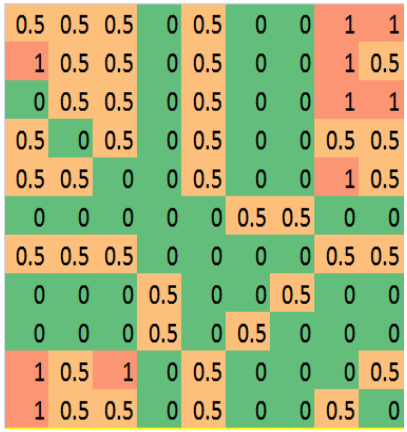
A.4.2 Delay Matrix

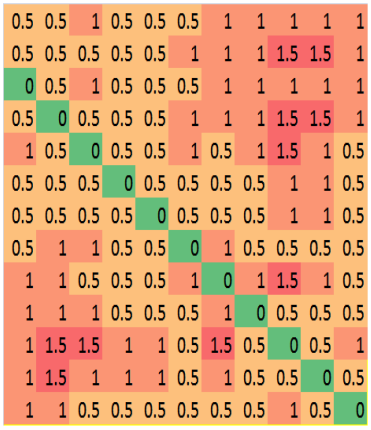


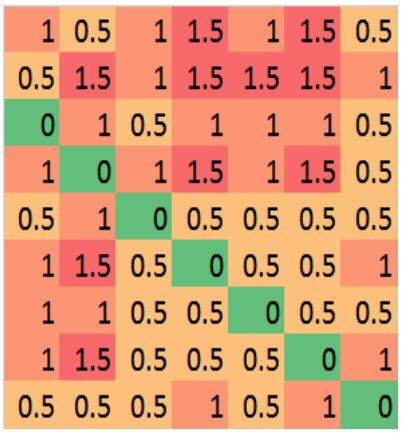




A.4.3 Energy Matrix







Appendix B : Matlab Code

B.1 Initialization

```
Close all;clear all;clc;
Sensornumber=600;
Actuaternumber=3;
Xmin = 0; %alwayas the case... Dont change this value
Xmax = 2000; % changeble1000
Ymin = 0; %alwayas the case... Dont change this value
Ymax = 2000; %changeble600
CTR=100;%100
Runnum=0;

Locationfile='location.mat';
If runnum==0
  X=Xmax*rand(1,sensornumber);
  Y=Ymax*rand(1,sensornumber);
  Scoordinates=[x';y'];
  Save(locationfile,'x','y');
Else
  Load(locationfile);
End

% ploting the location of the sensors with green squares
Plot(x,y,'rs','linewidth',2,...
  'Markeredgecolor','k',...
  'Markerfacecolor','g',...
  'Markersize',7);
Hold;

% actors
Xa =Xmax*rand(actuaternumber,1);
Ya=Ymax*rand(actuaternumber,1);

% Run Script of actors area divsion
 Areadiv;
```

B.2 Area Division Algorithm

```
Xa_ini=xa;
Ya_ini=ya;


% Calculating the centers of the sub areas
Xc = zeros(actuaternumber,1);
Yc = zeros(actuaternumber,1);
% to calculate the center of the divided areas
Areaboundary=[Xmin,Xmax,Ymin,Ymax]; %Xmin,Xmax,Ymin,Ymax
Actuaternumber;  % number of Actuaters
Newareacenter=[];  %initialization of the centers of the new clusters or
divided area.
Newareacenter=dividarea(areaboundary,actuaternumber,newareacenter); %
iterative algorithm to calculate the centers of the new clusters
Axis(areaboundary);
Hold on;


% Moving the actors to sub area centers
Xc= newareacenter(:,1);
Yc=newareacenter(:,2);
For i = 1:actuaternumber
  For j = 1:actuaternumber
    Fromcenter(i,j) =sqrt((xa(i)-xc(j)).^2 + (ya(i)-yc(j)).^2 );
  End
End

Xa=[];
Ya=[];
For i = 1:actuaternumber
  For j = 1:actuaternumber
    If(fromcenter(i,j) <= min(fromcenter(i,:)))
      Visited_j=j;
      Xa(i)=xc(j);
      Ya(i)=yc(j);
    End
  End
  Fromcenter(:,visited_j)=max(max(fromcenter));
End
```

B.3 Random Algorithm

```
% Random run
Nodesx=[x xa_ini'];% All the nodes= sensors + actors x location
Nodesy=[y ya_ini'];% All the nodes= sensors + actors y location

% Step 1 : Calculate the network distance matrix
Rnetwork_Dist_Mat =zeros(length(nodesx),length(nodesx));
For i = 1:length(nodesx)
  For j = 1:length(nodesx)
    Rnetwork_Dist_Mat(i,j) =sqrt((nodesx(i)-nodesx(j)).^2 + (nodesy(i)-
nodesy(j)).^2 );
  End
End

% Step 2 : Calculate the network adjacency matrix
Rnetwork_Adj_Mat =zeros(length(nodesx),length(nodesx));
  For i = 1:length(nodesx)
    For j = 1:length(nodesx)
         If ((rnetwork_Dist_Mat(i,j) <= CTR))
           Plot(nodesx(i),nodesy(i),nodesx(j),nodesy(j),'r');
        Rnetwork_Adj_Mat(i,j)=1;
        Else
         Rnetwork_Adj_Mat(i,j)=0;
        End
      End
  End

% Step 3 : Calculte the all pair shortest path (FW algorithm)

 Random_FW= floyd_warshall_all_sp(sparse(rnetwork_Adj_Mat));


% Step 4 : Calculte the enrgy cost of the network
% transciever works at full power

Rconsumedenergy = Random_FW * 1.2; %ujoules




% Step 5 : Plotting the network
    For i = 1:length(nodesx)
  For j = 1:length(nodesx)
    If ((rnetwork_Adj_Mat(i,j)==1))
      Plot([nodesx(i),nodesx(j)],[nodesy(i),nodesy(j)],'g');
    End
  End
End
```

B.4 Uniform Algorithm

```
% Uniform run
Nodesx=[x xc']';% All the nodes= sensors + actors x location
Nodesy=[y yc']';% All the nodes= sensors + actors y location

% Step 1 : Calculate the network distance matrix
Unetwork_Dist_Mat =zeros(length(nodesx),length(nodesx));
For i = 1:length(nodesx)
  For j = 1:length(nodesx)
    Unetwork_Dist_Mat(i,j) =sqrt((nodesx(i)-nodesx(j)).^2 + (nodesy(i)-
nodesy(j)).^2 );
  End
End

% Step 2 : Calculate the network adjacency matrix
Unetwork_Adj_Mat =zeros(length(nodesx),length(nodesx));
  For i = 1:length(nodesx)
    For j = 1:length(nodesx)
         If ((unetwork_Dist_Mat(i,j) <= CTR))
           Plot(nodesx(i),nodesy(i),nodesx(j),nodesy(j),'r');
        Unetwork_Adj_Mat(i,j)=1;
        Else
         Unetwork_Adj_Mat(i,j)=0;
        End
      End
  End

% Step 3 : Calculte the all pair shortest path (FW algorithm)

 Uniform_FW= floyd_warshall_all_sp(sparse(unetwork_Adj_Mat));


% Step 4 : Calculte the enrgy cost of the network
% transciever works at full power

Uconsumedenergy = Uniform_FW * 1.2;%ujoules



% Step 5 : Plotting the network
    For i = 1:length(nodesx)
  For j = 1:length(nodesx)
    If ((unetwork_Adj_Mat(i,j)==1))
      Plot([nodesx(i),nodesx(j)],[nodesy(i),nodesy(j)],'g');
    End
  End
End
```

B.5 COLA Algorithm

```
% COLA
% step 1 : find distance matrix from each sensor to each cluster center

Clusters = zeros(sensornumber,actuaternumber);
For i = 1:sensornumber
  For j = 1:actuaternumber
    Clusters(i,j) =sqrt((x(i)-xc(j)).^2 + (y(i)-yc(j)).^2 );
  End
End

% step 2 : find to which center the sensor is closest by making the
% corresponding entry 1 and the rest zeros

Simpleclusters = zeros(sensornumber,actuaternumber);

For i = 1:sensornumber
  For j = 1:actuaternumber
    If (Clusters(i,j) <= min(Clusters(i,:)))
      Simpleclusters(i,j)=1;
    End
  End
End

% step 3 :create a 3 diminsional matrix to save the coordinates of each
% cluster corresponding sensors

A3dclusters = zeros(sensornumber,2,actuaternumber);
For k = 1:actuaternumber
  For i = 1:sensornumber
    For j = 1:actuaternumber
      If (simpleclusters(i,k) == 1)
        A3dclusters(i,1,k)=x(i);
        A3dclusters(i,2,k)=y(i);
      End
    End
  End
End

% step 4 : create a cordinate vector for each cluster that will contain the
% coordinates of the sensors and the cenetr of that cluster
% this will show links for sensors close enough to the centers
Matrix=zeros(1,3);
Matrix(1,:)=[];
For k = 1:actuaternumber
  For i = 1:sensornumber
    For j = 1:actuaternumber
      If ((simpleclusters(i,k) == 1))
        New_row =[k a3dclusters(i,1,k) a3dclusters(i,2,k)];
        Matrix = [matrix; new_row];
        If (Clusters(i,j) <= CTR)
```

```
%
plot([a3dclusters(i,1,k),newareacenter(k,1)],[a3dclusters(i,2,k),newareacente
r(k,2)],'r');
        End
      End
    End
  End
End
% to eliminate the repeated rows in matrix

Clusterscord =unique(matrix,'rows');

%step 5 : find out the adjecency matrix for each cluster

Temp{1} =zeros(1,3);
For j=1: actuaternumber
  For i= 1:sensornumber
    If(clusterscord(i,1)==j)
      Temp{j}(i,1)=clusterscord(i,1);
      Temp{j}(i,2)=clusterscord(i,2);
      Temp{j}(i,3)=clusterscord(i,3);
    End
  End
End

% delete Zero raws in each cluster cordinate vector cell array

For i=1: actuaternumber
  Av=temp{i};
  Av(~any(temp{i},2),:)=[];
  Temp{i}=av;
  Temp{i}=[temp{i};[0,newareacenter(i,1),newareacenter(i,2)]];

End

%Step 6 : calculate the distance matrix for each point in the cluster to all
the other point within the cluster, including
%the center and save in a cell array

Cluster_Dist_Matrix{1} =zeros(1,actuaternumber);
For k =1 :actuaternumber
  [dim dimr] =size(temp{k});
  For i=1:dim
    For j=1:dim
      Cluster_Dist_Matrix{k}(i,j) =sqrt((temp{k}(i,2)-temp{k}(j,2)).^2 +
(temp{k}(i,3)-temp{k}(j,3)).^2 );
      If (Cluster_Dist_Matrix{k}(i,j) ==0)
        Cluster_Dist_Matrix{k}(i,j)=inf;
      End
    End
  End
End

% Step 7 : creating the adjacency matrix
```

```
Cluster_Adjacy_Matrix{1} =zeros(1,actuaternumber);
For k =1 :actuaternumber
  [dim dimr] =size(temp{k});
  For i=1:dim
    For j=1:dim
      If ((Cluster_Dist_Matrix{k}(i,j) <= min(Cluster_Dist_Matrix{k}(i,:)))
|| ((Cluster_Dist_Matrix{k}(i,j) <= CTR)))
        Cluster_Adjacy_Matrix{k}(i,j)=1;
              Else
        Cluster_Adjacy_Matrix{k}(i,j)=0;
              End
    End
  End
End

%the first cluster has zero columns that must be deleted to have a squared
%matrix... The code below does that

[rows cols]= size (Cluster_Adjacy_Matrix{1});
For i= rows+1 : cols
  Cluster_Adjacy_Matrix{1}(:,rows)= [];
End

%step 8 : applying floyd warshall algorithm on the clusters adjacency
%matrices

[dim dimr] =size(temp{k});
Ftemp{1} =zeros(1,actuaternumber);
For k=1:actuaternumber
  Ftemp{k}= floyd_warshall_all_sp(sparse(Cluster_Adjacy_Matrix{k}));
End

%step 9 : find the maxlist
Maxlist{1} =zeros(1,actuaternumber);
For k=1:actuaternumber
  [dim1 xtrash] =size(temp{k});
  For i=1:dim1
    Maxlist{k}(i,2) = max(Ftemp{k}(i,:));
    Maxlist{k}(i,1) = i;
  End
End

%step 10 : find the mincost
Mincost{1} =zeros(2,actuaternumber);
For k=1:actuaternumber
  [valuemincost locationmincost]=min(maxlist{k}(:,2));
  Mincost{k}=maxlist{k}(locationmincost(1),:);
  Valuemincost=[];
  Locationmincost=[];
End

For k=1:actuaternumber
  Algocenter{k}=[temp{k}(mincost{k}(1),2) temp{k}(mincost{k}(1),3)];
  Save('algocenter.mat','algocenter');
  Algocenterplot(k,:)=[algocenter{k}(:)];
```

```
End

%Step 11: Add the new location to the clusters and recalculate

Temp1{1} =zeros(1,3);
For j=1: actuaternumber
  For i= 1:sensornumber
    If(clusterscord(i,1)==j)
      Temp1{j}(i,1)=clusterscord(i,1);
      Temp1{j}(i,2)=clusterscord(i,2);
      Temp1{j}(i,3)=clusterscord(i,3);
    End
  End
End
For i=1: actuaternumber
  Av=temp1{i};
  Av(~any(temp1{i},2),:)=[];
  Temp1{i}=av;
  Temp1{i}=[temp1{i};200,algocenterplot(i,1),algocenterplot(i,1)];

End

Cluster_Dist_Matrix1{1} =zeros(1,actuaternumber);
For k =1 :actuaternumber
  [dim dimr] =size(temp1{k});
  For i=1:dim
    For j=1:dim
      Cluster_Dist_Matrix1{k}(i,j) =sqrt((temp1{k}(i,2)-temp1{k}(j,2)).^2 +
(temp1{k}(i,3)-temp1{k}(j,3)).^2 );
      If (Cluster_Dist_Matrix1{k}(i,j) ==0)
        Cluster_Dist_Matrix1{k}(i,j)=inf;
      End
    End
  End
End



Cluster_Adjacy_Matrix1{1} =zeros(1,actuaternumber);
For k =1 :actuaternumber
  [dim dimr] =size(temp1{k});
  For i=1:dim
    For j=1:dim
      If ((Cluster_Dist_Matrix1{k}(i,j) <= min(Cluster_Dist_Matrix1{k}(i,:)))
|| ((Cluster_Dist_Matrix1{k}(i,j) <= CTR)))
        Cluster_Adjacy_Matrix1{k}(i,j)=1;
              Else
        Cluster_Adjacy_Matrix1{k}(i,j)=0;
              End
    End
  End
End
```

```
[rows cols]= size (Cluster_Adjacy_Matrix1{1});
For i= rows+1 : cols
  Cluster_Adjacy_Matrix1{1}(:,rows)= [];
End



[dim dimr] =size(temp1{k});
Ftemp1{1} =zeros(1,actuaternumber);
For k=1:actuaternumber
  Ftemp1{k}= floyd_warshall_all_sp(sparse(Cluster_Adjacy_Matrix1{k}));
End

% Step 12 : Calculte the enrgy cost of the network
% transciever works at full power
 Colaconsumedenergy{1} =zeros(1,actuaternumber);
 For k =1 :actuaternumber
  [dim dimr] =size(temp1{k});
  For i=1:dim
    For j=1:dim
      If ((Cluster_Dist_Matrix1{k}(i,j) <= min(Cluster_Dist_Matrix1{k}(i,:)))
|| ((Cluster_Dist_Matrix1{k}(i,j) <= CTR)))
       Colaconsumedenergy{k}(i,j) = Ftemp1{k}(i,j) * 1.2;%ujoules ; %ujoules
      Else
        Colaconsumedenergy{k}(i,j) =0;
      End
    End
  End
End
 For k=1:actuaternumber
  Colaconsumedenergy_ASP{k}=
floyd_warshall_all_sp(sparse(colaconsumedenergy{k}));
End
```

B.6 DEMA

```
% DEMA
% step 1 : Compute distance matrix

Clusters = zeros(SensorNumber,ActuaterNumber);
for i = 1:SensorNumber
    for j = 1:ActuaterNumber
        Clusters(i,j) =sqrt((x(i)-xc(j)).^2 + (y(i)-yc(j)).^2 );
    end
end

% step 2 : find to which center the sensor is closest by making the
% corresponding entry 1 and the rest zeros

simpleClusters = zeros(SensorNumber,ActuaterNumber);

for i = 1:SensorNumber
    for j = 1:ActuaterNumber
        if (Clusters(i,j) <= min(Clusters(i,:)))
            simpleClusters(i,j)=1;
        end
    end
end

% step 3 :create a 3 diminsional matrix to save the coordinates of each
% cluster corresponding sensors

A3dClusters = zeros(SensorNumber,2,ActuaterNumber);
for k = 1:ActuaterNumber
    for i = 1:SensorNumber
        for j = 1:ActuaterNumber
            if (simpleClusters(i,k) == 1)
                A3dClusters(i,1,k)=x(i);
                A3dClusters(i,2,k)=y(i);
            end
        end
    end
end

% step 4 : create a cordinate vector for each cluster that will contain the
% coordinates of the sensors and the cenetr of that cluster
% this will show links for sensors close enough to the centers
matrix=zeros(1,3);
matrix(1,:)=[];
for k = 1:ActuaterNumber
    for i = 1:SensorNumber
        for j = 1:ActuaterNumber
            if ((simpleClusters(i,k) == 1))
                new_row =[k A3dClusters(i,1,k) A3dClusters(i,2,k)];
                matrix = [matrix ; new_row];
                if (Clusters(i,j) <= CTR)
%
plot([A3dClusters(i,1,k),NewAreaCenter(k,1)],[A3dClusters(i,2,k),NewAreaCente
r(k,2)],'r') ;
```

```
                end
            end
        end
    end
end
% to eliminate the repeated rows in matrix

ClustersCord =unique(matrix,'rows');

%step 5 : find out the adjecency matrix for each cluster

temp{1} =zeros(1,3);
for j=1: ActuaterNumber
    for i= 1:SensorNumber
        if(ClustersCord(i,1)==j)
            temp{j}(i,1)=ClustersCord(i,1);
            temp{j}(i,2)=ClustersCord(i,2);
            temp{j}(i,3)=ClustersCord(i,3);
        end
    end
end

% delete Zero raws in each cluster cordinate vector cell array

for i=1: ActuaterNumber
    av=temp{i};
    av(~any(temp{i},2),:)=[];
    temp{i}=av;
    temp{i}=[temp{i};[0,NewAreaCenter(i,1),NewAreaCenter(i,2)]];

end

%Step 6 : calculate the distance matrix for each point in the cluster to all
the other point within the cluster, including
%the center and save in a cell array

Cluster_Dist_Matrix{1} =zeros(1,ActuaterNumber);
for k =1 :ActuaterNumber
    [dim dimr] =size(temp{k});
    for i=1:dim
        for j=1:dim
            Cluster_Dist_Matrix{k}(i,j) =sqrt((temp{k}(i,2)-temp{k}(j,2)).^2
+ (temp{k}(i,3)-temp{k}(j,3)).^2 );
            if (Cluster_Dist_Matrix{k}(i,j) ==0)
                Cluster_Dist_Matrix{k}(i,j)=inf;
            end
        end
    end
end

% Step 7 : creating the adjacency matrix

Cluster_Adjacy_Matrix{1} =zeros(1,ActuaterNumber);
for k =1 :ActuaterNumber
    [dim dimr] =size(temp{k});
```

```
    for i=1:dim
        for j=1:dim
            if ((Cluster_Dist_Matrix{k}(i,j) <=
min(Cluster_Dist_Matrix{k}(i,:))) || ((Cluster_Dist_Matrix{k}(i,j) <= CTR)))
                Cluster_Adjacy_Matrix{k}(i,j)=1;
                            else
                Cluster_Adjacy_Matrix{k}(i,j)=0;
                           end
        end
    end
end

%the first cluster has zero columns that must be deleted to have a squared
%matrix... the code below does that

[rows cols]= size (Cluster_Adjacy_Matrix{1});
for i= rows+1 : cols
    Cluster_Adjacy_Matrix{1}(:,rows)= [];
end

%step 5 :  applying flyd warshall algorithm on the clusters adjacency
%matices
[dim dimr] =size(temp{k});
Ftemp{1} =zeros(1,ActuaterNumber);
for k=1:ActuaterNumber
    Ftemp{k}= floyd_warshall_all_sp(sparse(Cluster_Adjacy_Matrix{k}));
end
% n1
for k =1 :ActuaterNumber
    [dim dimr] =size(temp{k});
    for i=1:dim
        for j=1:dim
            if (Ftemp{k}(i,j) == 1)
%
plot([temp{k}(i,2),temp{k}(j,2)],[temp{k}(i,3),temp{k}(j,3)],'g') ;
            end
        end
    end
end

Cluster_Dist{1} =zeros(1,ActuaterNumber);
for k =1 :ActuaterNumber
    [dim dimr] =size(temp{k});
    for i=1:dim
        for j=1:dim
            Cluster_Dist{k}(i,j) =sqrt((temp{k}(i,2)-temp{k}(j,2)).^2 +
(temp{k}(i,3)-temp{k}(j,3)).^2 );

        end
    end
end

% her we can get the number of ones...tows in the ftemp
stats{1}= zeros(1,ActuaterNumber);
for k=1:ActuaterNumber
        stats{k}=histc(Ftemp{k},1:10);
```

```
end

Average{1} =zeros(1,ActuaterNumber);
MaxOneStats{1} =zeros(1,ActuaterNumber);
MaxTwoStats{1} =zeros(1,ActuaterNumber);
for k=1:ActuaterNumber
    [dim1 xtrash] =size(temp{k});
    for i=1:dim1-1
       Average{k} = mean(Ftemp{k});
       MaxOneStats{k}= max(stats{k}(1,:));
       MaxTwoStats{k}= max(stats{k}(2,:));

    end
end
%step 2:  find the Min Average
MinAve{1} =zeros(1,ActuaterNumber);
MinOS{1} =zeros(1,ActuaterNumber);
MinTS{1} =zeros(1,ActuaterNumber);
for k=1:ActuaterNumber
    [dim1 xtrash] =size(temp{k});
    for j =1:dim1
        MinAve{k}(2,j) =(Average{k}(:,j));
        MinAve{k}(1,j) = j ;
        MinOS{k}(2,j) = (stats{k}(1,j));
        MinOS{k}(1,j) =j;
        MinTS{k}(2,j) = (stats{k}(2,j));
        MinTS{k}(1,j) =j;

    end
end

MAver{1} =zeros(1,1);
MinOSr{1} =zeros(1,1);
MinTSr{1} =zeros(1,1);
for k=1:ActuaterNumber

        MAver{k}(2,1) =  min(MinAve{k}(2,:));
        MinOSr{k}(2,1) = MaxOneStats{k};
        MinTSr{k}(2,1) = MaxTwoStats{k};
       [dim1 xtrash] =size(temp{k});
       for j =1:dim1
           if MAver{k}(2,1)== min(MinAve{k}(2,j))
        MAver{k}(1,1) = j ;
           end

             if MinOSr{k}(2,1)== (MinOS{k}(2,j))
        MinOSr{k}(1,1) = j ;
             end

               if MinTSr{k}(2,1)== (MinTS{k}(2,j))
        MinTSr{k}(1,1) = j ;
           end
       end
end
```

```
% find the algo center
for k=1:ActuaterNumber
AlgoCenter_X{k}
=(temp{k}(MAver{k}(1,1),2)+temp{k}(MinOSr{k}(1,1),2)+temp{k}(MinTSr{k}(1,1),2
))/3;
AlgoCenter_Y{k}=(temp{k}(MAver{k}(1,1),3)+temp{k}(MinOSr{k}(1,1),3)+temp{k}(M
inTSr{k}(1,1),3))/3;
end
% assign the algo center to next run
for k=1:ActuaterNumber
    AlgoCenter{k}=[AlgoCenter_X{k} AlgoCenter_Y{k}];
    save('AlgoCenter.mat','AlgoCenter');
    AlgoCenterplot(k,:)=[AlgoCenter{k}(:)];
end

plot(AlgoCenterplot(:,1),AlgoCenterplot(:,2),'s','LineWidth',2,...
    'MarkerEdgeColor','k',...
    'MarkerFaceColor','R',...
    'MarkerSize',10);
Results= zeros(1,1);
for k=1:ActuaterNumber
    [dim1 xtrash] =size(temp{k});
    Results(k,1) = k;
       Results(k,2) = mean(Ftemp{k}(dim1,:));

end

%Recalculate everything with the new location
%=========================================

temp2{1} =zeros(1,3);
for j=1: ActuaterNumber
    for i= 1:SensorNumber
        if(ClustersCord(i,1)==j)
            temp2{j}(i,1)=ClustersCord(i,1);
            temp2{j}(i,2)=ClustersCord(i,2);
            temp2{j}(i,3)=ClustersCord(i,3);
        end
    end
end

% delete Zero raws in each cluster cordinate vector cell array

for i=1: ActuaterNumber
    av=temp2{i};
    av(~any(temp2{i},2),:)=[];
    temp2{i}=av;
    temp2{i}=[temp2{i};300,AlgoCenterplot(i,1),AlgoCenterplot(i,1)];

end

%Step 6 : calculate the distance matrix for each point in the cluster to all
the other point within the cluster, including
%the center and save in a cell array
```

```
Cluster_Dist_Matrix2{1} =zeros(1,ActuaterNumber);
for k =1 :ActuaterNumber
    [dim dimr] =size(temp2{k});
    for i=1:dim
        for j=1:dim
            Cluster_Dist_Matrix2{k}(i,j) =sqrt((temp2{k}(i,2)-
temp2{k}(j,2)).^2 + (temp2{k}(i,3)-temp2{k}(j,3)).^2 );
            if (Cluster_Dist_Matrix2{k}(i,j) ==0)
                Cluster_Dist_Matrix2{k}(i,j)=inf;
            end
        end
    end
end

% Step 7 : creating the adjacency matrix

Cluster_Adjacy_Matrix2{1} =zeros(1,ActuaterNumber);
for k =1 :ActuaterNumber
    [dim dimr] =size(temp2{k});
    for i=1:dim
        for j=1:dim
            if ((Cluster_Dist_Matrix2{k}(i,j) <=
min(Cluster_Dist_Matrix2{k}(i,:))) || ((Cluster_Dist_Matrix2{k}(i,j) <=
CTR)))
                Cluster_Adjacy_Matrix2{k}(i,j)=1;
                           else
                Cluster_Adjacy_Matrix2{k}(i,j)=0;
                          end
        end
    end
end

%the first cluster has zero columns that must be deleted to have a squared
%matrix... the code below does that

[rows cols]= size (Cluster_Adjacy_Matrix2{1});
for i= rows+1 : cols
    Cluster_Adjacy_Matrix2{1}(:,rows)= [];
end

%step 5 :  applying flyd warshall algorithm on the clusters adjacency
%matices
[dim dimr] =size(temp2{k});
Ftemp2{1} =zeros(1,ActuaterNumber);
for k=1:ActuaterNumber
    Ftemp2{k}= floyd_warshall_all_sp(sparse(Cluster_Adjacy_Matrix2{k}));
end
% n1
for k =1 :ActuaterNumber
    [dim dimr] =size(temp2{k});
    for i=1:dim
        for j=1:dim
            if (Ftemp2{k}(i,j) == 1)
%
plot([temp2{k}(i,2),temp2{k}(j,2)],[temp2{k}(i,3),temp2{k}(j,3)],'g') ;
            end
```

```
        end
    end
end

% Calculte the energy consumption of the algorithm :

 AlgoEnergyMatrix{1} =zeros(1,ActuaterNumber);
 for k =1 :ActuaterNumber
    [dim dimr] =size(temp2{k});
    for i=1:dim
        for j=1:dim
            if ((Cluster_Dist_Matrix2{k}(i,j) <=
min(Cluster_Dist_Matrix2{k}(i,:))) || ((Cluster_Dist_Matrix2{k}(i,j) <=
CTR)))
              AlgoEnergyMatrix{k}(i,j) =  Cluster_Dist_Matrix2{k}(i,j)
*1.2/200; %ujoules
            else
               AlgoEnergyMatrix{k}(i,j) =0;
            end
        end
    end
end
 for k=1:ActuaterNumber
    AlgoEnergyMatrix_ASP{k}=
floyd_warshall_all_sp(sparse(AlgoEnergyMatrix{k}));
end
```

References

[1] Akyildiz, Ian F., Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. "Wireless sensor networks: a survey." Computer networks 38, no. 4 (2002): 393-422.

[2] Bilstrup, Urban, K. Sjoberg, Bertil Svensson, and P-A. Wiberg. "Capacity limitations in wireless sensor networks." In Emerging Technologies and Factory Automation, 2003. Proceedings. ETFA'03. IEEE Conference, vol. 1, pp. 529-536. IEEE, 2003.

[3] Younis, Mohamed, and Kemal Akkaya. "Strategies and techniques for node placement in wireless sensor networks: A survey." Ad Hoc Networks 6, no. 4 (2008): 621-655.

[4] Akkaya, Kemal, and Mohamed Younis. "A survey on routing protocols for wireless sensor networks." Ad hoc networks 3, no. 3 (2005): 325-349.

[5] Akkaya, Kemal, and Mohamed Younis. "COLA: A coverage and latency aware actor placement for wireless sensor and actor networks." In Vehicular Technology Conference, 2006. VTC-2006 Fall. 2006 IEEE 64th, pp. 1-5. IEEE, 2006.

[6] Akkaya, Kemal, and Mohamed Younis. "C2AP: Coverage-aware and connectivity-constrained actor positioning in wireless sensor and actor networks." In Performance, Computing, and Communications Conference, 2007. IPCCC 2007. IEEE Internationa, pp. 281-288. IEEE, 2007.

[7] Akkaya, Kemal, and Mohamed Younis. "Coverage and latency aware actor placement mechanisms in WSANs." International Journal of Sensor Networks 3, no. 3 (2008): 152-1 64.

[8] Akkaya, Kemal, and Mohamed Younis. "COLA: A coverage and latency aware actor placement for wireless sensor and actor networks." In Vehicular Technology Conference, 2006. VTC-2006 Fall. 2006 IEEE 64th, pp. 1-5. IEEE, 2006.

[9] Labrador, Miguel A., and Pedro M. Wightman. Topology Control in Wireless Sensor Networks: with a companion simulation tool for teaching and research. Springer, 2009.

[10] Raghavendra, Cauligi S., Krishna M. Sivalingam, and Taieb Znati, eds. Wireless sensor networks. Kluwer Academic Pub, 2004.

[11] Karl, Holger, and Andreas Willig. Protocols and architectures for wireless sensor networks. Wiley-Interscience, 2007.

[12] Sohraby, Kazem, Daniel Minoli, and Taieb Znati. Wireless sensor networks: technology, protocols, and applications. Wiley-interscience, 2007.

[13] BBN Technologies, "Boomerang shooter detection system". Available at http://bbn.com/boomerang, (acceessed on 13 may 2013) .

[14] Basha, Elizabeth A., Sai Ravela, and Daniela Rus. "Model-based monitoring for early warning flood detection." In Proceedings of the 6th ACM conference on Embedded network sensor systems, pp. 295-308. ACM, 2008.

[15] U.S Department of Energy "Artificial Retina project". Available at http://artificialretina.energy.gov, (acceessed on 13 may 2013) .

[16] Kim, Younghun, Thomas Schmid, Zainul M. Charbiwala, Jonathan Friedman, and Mani B. Srivastava. "NAWMS: nonintrusive autonomous water monitoring system." In Proceedings of the 6th ACM conference on Embedded network sensor systems, pp. 309-322. ACM, 2008.

[17] Petrushin, Valery A., Gang Wei, Omer Shakil, Damian Roqueiro, and V. Gershman. "Multiple-sensor indoor surveillance system." In Computer and Robot Vision, 2006. The 3rd Canadian Conference on, pp. 40-40. IEEE, 2006.

[18] Krishnamurthy, Lakshman, Robert Adler, Phil Buonadonna, Jasmeet Chhabra, Mick Flanigan, Nandakishore Kushalnagar, Lama Nachman, and Mark Yarvis. "Design and deployment of industrial sensor networks: experiences from a semiconductor plant and the north sea." In Proceedings of the 3rd international conference on Embedded networked sensor systems, pp. 64-75. ACM, 2005.

[19] Paek, Jeongyeup, Krishna Chintalapudi, John Caffrey, Ramesh Govindan, and Sami Masri. "A wireless sensor network for structural health monitoring: Performance and experience." (2005).

[20] Mechitov, Kirill, W. Kim, G. Agha, and T. Nagayama. "High-frequency distributed sensing for structure monitoring." In Proc. First Intl. Workshop on Networked Sensing Systems (INSS 04). 2004.

[21] Berry, Jonathan W., Lisa Fleischer, William E. Hart, Cynthia A. Phillips, and Jean-Paul Watson. "Sensor placement in municipal water networks." Journal of Water Resources Planning and Management 131, no. 3 (2005): 237-243.

[22] Xu, Kenan, Hossam Hassanein, Glen Takahara, and Quanhong Wang. "Relay node deployment strategies in heterogeneous wireless sensor networks: single-hop communication case." In Global Telecommunications Conference, 2005. GLOBECOM'05. IEEE, vol. 1, pp. 5-pp. 2005.

[23] Xu, Kenan, Hossam Hassanein, Glen Takahara, and Q. Wang. "Relay node deployment strategies in heterogeneous wireless sensor networks: multiple-hop communication case." In Proc. IEEE Secon, pp. 575-585. 2005.

[24] Huang, Chi-Fu, and Yu-Chee Tseng. "The coverage problem in a wireless sensor network." Mobile Networks and Applications 10, no. 4 (2005): 519-528.

[25] Boukerche, A., Xin Fei, and Regina B. Araujo. "A coverage preserving and fault tolerant based scheme for irregular sensing range in wireless sensor networks." In the Proceedings of the 49 Annual th IEEE Global Communication Conference (Globecom'06), San Francisco, CA. 2006.

[26] S. Meguerdichian, F. Koushanfar, M. Potkonjak, M.B. Srivastava, Coverage problems in wireless ad-hoc sensor networks, in: Proceedings of the 20th International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'01), Anchorage, Alaska, April 2001

[27] Kar, Koushik, and Suman Banerjee. "Node placement for connected coverage in sensor networks." In WiOpt'03: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks. 2003.

[28] Bredin, Jonathan L., Erik D. Demaine, MohammadTaghi Hajiaghayi, and Daniela Rus. "Deploying sensor networks with guaranteed capacity and fault tolerance." In Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing, pp. 309-319. ACM, 2005.

[29] Toumpis, Stavros, and Leandros Tassiulas. "Packetostatics: Deployment of massively dense sensor networks as an electrostatics problem." In INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE, vol. 4, pp. 2290-2301. IEEE, 2005.

[30] Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to algorithms. MIT press, 2001.

[31] Cherkassky, Boris V., Andrew V. Goldberg, and Tomasz Radzik. "Shortest paths algorithms: theory and experimental evaluation." Mathematical programming 73, no. 2 (1996): 129-174.

[32] Zhou, Hai-Ying, Dan-Yan Luo, Yan Gao, and De-Cheng Zuo. "Modeling of node energy consumption for wireless sensor networks." Wireless Sensor Network 3, no. 1 (2011): 18-23.

[33] Tang, Jian, Bin Hao, and Arunabha Sen. "Relay node placement in large scale wireless sensor networks." Computer communications 29, no. 4 (2006): 490-501.

[34] Hou, Y. Thomas, Yi Shi, Hanif D. Sherali, and Scott F. Midkiff. "On energy provisioning and relay node placement for wireless sensor networks." Wireless Communications, IEEE Transactions on 4, no. 5 (2005): 2579-2590.

[35] Oyman, E. Ilker, and Cem Ersoy. "Multiple sink network design problem in large scale wireless sensor networks." In Communications, 2004 IEEE International Conference on, vol. 6, pp. 3663-3667. IEEE, 2004.

[36] Youssef, Waleed, and Mohamed Younis. "Intelligent gateways placement for reduced data latency in wireless sensor networks." In Communications, 2007. ICC'07. IEEE International Conference on, pp. 3805-3810. IEEE, 2007.

[37] Gandham, Shashidhar Rao, Milind Dawande, Ravi Prakash, and Subbarayan Venkatesan. "Energy efficient schemes for wireless sensor networks with multiple mobile base stations." In Global telecommunications conference, 2003. GLOBECOM'03. IEEE, vol. 1, pp. 377-381. IEEE, 2003.

[38] Pan, Jianping, Y. Thomas Hou, Lin Cai, Yi Shi, and Sherman X. Shen. "Locating base-stations for video sensor networks." In Vehicular Technology Conference, 2003. VTC 2003-Fall. 2003 IEEE 58th, vol. 5, pp. 3000-3004. IEEE, 2003.