Distributed and Localized Strategies for Energy Restoration in Wireless Sensor Networks

by

Elio Velazquez

A *thesis submitted to*
the Faculty of Graduate and Postdoctoral Affairs
in partial fulfillment of
the requirements for the degree of

DOCTOR OF PHILOSOPHY

Ottawa-Carleton Institute for Computer Science
School of Computer Science
Carleton University
Ottawa, Ontario

February, 2011

Your file Votre référence
ISBN: 978-0-494-81554-0
Our file Notre référence
ISBN: 978-0-494-81554-0

Abstract

Energy management is one of the main hurdles in the quest for autonomous and reliable Wireless Sensor Networks (WSN). This thesis examines several scenarios where mobility is added to the network components in order to carry out energy management tasks. The main goal is to increase network availability by recharging or redeploying "depleted" sensors with the help of mobile entities.

For static sensors networks we provide a cluster-based solution where (1) workload is balanced, (2) the movement of the maintenance entities is minimized and (3) there is a small number of sensor communications. This problem is a variant of the so-called Facility Location Problem (FLP). Although finding the optimal network partition and placement of the facilities is a NP-hard problem, we show a simple and efficient solution that provides partitions of remarkable quality. The experimental analysis shows that sensor communication cost remains low as the size of the network increases, there is a rapid progression towards convergence and the quality of the final partition is similar to centralized clustering benchmarks.

We also study a particular instance of the Frugal Feeding Problem (FFP), where mobility becomes a sensor's attribute and service stations are static. In this scenario, we examine the mobility strategies, underlying topologies and network parameters that guarantee an autonomous sensor recharge. The experimental results show that taking a proactive approach to energy redistribution and network fatigue outperforms passive strategies. The greedy closest-first swapping-based mobility strategy offered the best overall performance among all the proactive approaches studied and the proposed Compass Directed Graph provides a good underlying topology to achieve energy equilibrium.

Acknowledgements

I would like to thank my thesis supervisor Dr. Nicola Santoro for his scientific guidance and support throughout this research. Many thanks to Dr. Mark Lanthier for his collaboration, comments and valuable help with the experiments.

I would like to express my appreciation to my colleagues from OIRP. In particular, to Mr. Ian Calvert and Mr. Bruce Winer for their support and constant encouragement. Thank you Bruce for always being ready to listen and help.

I am indebted to my parents for their love and wonderful example of dedication. Most importantly, I owe my immense gratitude to my kids and especially to my wife Yoli, for her energy, enthusiasm and patience.

Gracias.

Table of Contents

List of Figures

List of Acronyms

WSN	Wireless Sensor Networks
MSN	Mobile Sensor Networks
NAMOS	Networked Aquatic Microbial Observing System
BMS	Balanced Maintenance System
LEACH	Low Energy Adaptive Clustering Hierarchy
HEED	Hybrid Energy-Efficient Distributed Clustering
HAC	Hierarchical Agglomerative Clustering
FIFO	First-in-First-out
FLP	Facility Location Problem
FFP	Frugal Feeding Problem
GFG	Greedy-Face-Greedy
CDG	Compass Directed Graph
CDGG	Compass Directed Gabriel Graph
CDRNG	Compass Directed Relative Neighbor Graph

Table 1: List of acronyms used throughout this dissertation

Chapter 1

Introduction

1.1 Overview

Wireless sensor networks (WSN) as a new stream of wireless ad-hoc networks have gained popularity in recent years. Despite being application specific and inheriting most of the constraints in computation, communications and energy, their applications are extending far beyond the classical security or surveillance use. As sensor technology becomes more affordable and reliable, the possibility of deploying hundreds or even thousands of nodes gets closer to reality. Sensor networks are being studied and deployed in a variety of fields, from security and environmental/wildlife monitoring to industrial and medical applications. Examples of these applications can be found in (e.g., [1, 26, 43, 52, 55, 65, 67, 72]).

Current efforts in sensor network research focus on finding more efficient ways to deploy, redeploy or relocate sensors in the field while guaranteeing the desired levels of coverage (e.g. [14, 29, 51, 78, 82]). In general, the key areas being investigated by the research community include: deployment, relocation, routing, data dissemination and gathering, security and energy management. There is significant overlap among them and solutions for each of these areas should consider their impact on others.

Energy consumption is a constant concern in every aspect of a sensor network design. Regardless of the problem being addressed, the ultimate goal of any sensor network is to achieve accurate sensing and maximize lifetime while maintaining an

acceptable level of coverage. When there are thousands of nodes deployed, there is a need to efficiently maintain, repair or replace them either by humans or artificial entities. In any Wireless Sensor Network deployment, the sensors' batteries will eventually deplete and loss of coverage would occur. The most simplistic approach to cope with an eventual loss of coverage would be to deploy more sensors to compensate for the loss of the depleted ones; but for obvious environmental reasons this kind of solution is not sustainable. More creative approaches attempt to extract energy from the environment to extend network operating life (e.g., [58, 59]). Others explore the use of mobile entities (e.g., robots, actuators, service stations) in conjunction with clustering techniques (e.g., [39, 48, 66, 76]) as a means of saving energy and coordinating sensors for data gathering, aggregation and network repair (e.g., [48, 76]). In general, energy management strategies can be categorized into two groups with some degree of overlap: *cluster-based* approaches (e.g., [28, 45, 57, 83]) or *mobility-based* approaches (e.g., [39, 46, 49, 79, 80]). The idea of using robots in sensor networks has also been proposed for other network maintenance tasks (e.g., [35, 38]). In the first part of this thesis we are interested precisely in this approach.

In general, the problem of energy management and network repair involves the use of special entities. These entities, which could be particularly equipped sensors or dedicated maintenance robots, will recharge or replace depleted sensors. The maintenance entities should coordinate their actions in a distributed and scalable manner. The ultimate goal is to maximize network availability by avoiding permanent sensor losses due to battery depletion. Some of the existing solutions for energy management, with or without robots, rely on some kind of clustering or partitioning of the network and the use of the special entities as cluster heads. The energy management problem is basically addressed by either creating a fixed partition of the field or by

constructing and maintaining dynamic clustering structures which depend on the position of the cluster heads.

In scenarios where the recharging or replacing capacity of each maintenance robot is bounded and the same among all robots, the problem is to find a network partition where the robots act as cluster heads, and: (1) each robot minimizes the sum of the distances to all sensors in the cluster, and at the same time (2) the workload among the clusters is balanced. This problem should be solved efficiently at least for the weakest elements of the system, the sensors; that is (3) the number of sensor communications should be kept small. A simpler version of this problem is the Facility Location Problem (FLP), also studied in the context of sensor networks (e.g., [24, 37]), where a number of service facilities (i.e., robots) can be placed only in a subset of a predefined number of locations. The goal is to provide services to the sensors at minimal traveling cost. Finding an optimal placement of the facilities is a NP-hard problem [62]; hence our problem is also NP-hard.

The second part of the thesis deals with the scenario where mobility is added to the sensors and static recharge facilities are deployed throughout the sensing area. This could be seen as the opposite of the aforementioned problem, where the responsibility for maintaining the overall health of the network is shifted to the sensor side, whereas the service facilities play a more passive role. In this scenario, the service facilities are equipped with a fixed number of recharging sockets and the sensors should coordinate their actions to make an efficient use of this shared resource. Under normal circumstances and overall energy levels at an acceptable state, the problem lies in deciding whether the sensors act while their batteries are still fully operational or later, when their batteries reach a critical level (i.e., close to depletion). This work attempts to provide an answer to this question by addressing the problem of network

fatigue from a passive and a proactive perspective.

The idea of dynamic redistribution of the network, when it is still in a healthy state, seems promising. By exchanging positions with other sensors, lower energy sensors can get closer to the service stations and capture "front seats" for when it is time to recharge their batteries. However, this proactive behaviour introduces another problem. The sensors need to communicate and coordinate their actions in order to achieve a common goal relying only on local information. This extra need for coordination comes at a cost that should not overwhelm the entire system, preventing it from outperforming a passive approach. Furthermore, sensors should coordinate their moves in a loop-free manner so the intended destination (i.e., recharge facility) is reached in a finite number of moves. The ultimate goal of a proactive approach is to reach a state of equilibrium where there are no further sensor losses due to battery depletion. This work also examines some underlying topologies that guarantee a loop-free mobility strategy as well as the network parameters needed to achieve a state of energy equilibrium.

1.2 Contributions of the Thesis

This thesis examines one of the most challenging topics in wireless sensor network design: energy management and restoration. First, the problem domain is divided into two major categories based on the mobility property: 1) static sensors with mobile maintenance facilities and 2) mobile capable sensors with static maintenance facilities. For the static sensor scenario, this work proposed a balanced cluster-based energy management system combining a static cluster partitioning with a distance-aware robot positioning. The proposed model satisfies the condition of workload balance among all service facilities as well as minimizes total service travel time. This model is extended to solve the Facility Location Problem (FLP) in the sensor network domain. The proposed sensor swapping approach to the FLP is completely distributed and localized. There is no central entity with global knowledge, and the virtual sensor swapping operations take place only between adjacent areas.

Another contribution of this thesis is the development of prototypes and the verification of the proposed algorithms based on simulations and experimental results. The test results for the static sensor scenario showed that the cost in terms of sensor communications remained low as the network size increased. All the algorithms tested showed a fast progression towards convergence where after only a few iterations, the quality of the partition was almost identical to the one obtained at the convergence point. Furthermore, the quality of the partition achieved by the sensor selection algorithm and later by applying the sensor swapping optimization, was similar to the selected centralized clustering benchmarks (e.g., K-Means, HAC).

For the mobile sensor scenario, this work diverges from the existing research by evaluating passive vs. proactive strategies for energy restoration while maintaining

topological integrity. In summary, the second part of the thesis proposes to reduce the problem of recharging mobile sensors in a network of arbitrary topology to the implementation of proactive energy-aware mobility strategies applied to a logical Compass Directed Graph (CDG) constructed on top of the original deployment. The proposed graph is dynamic, self-correcting and provides a sound foundation for loop-free mobility strategies.

The proposed strategies for the mobile sensor scenario were also validated through a series of simulations, which explored several variables that impact the performance of the proactive solutions, such as topology, number of neighbours, size of the network, number of recharging sockets, sensor transmission range and locomotion costs. The test results showed that all the pro-active strategies analyzed reached the state of energy equilibrium. The number of graph neighbours (i.e., node degree) had a positive impact on the cumulative number of sensor losses until equilibrium, proving that for networks with 100:1 sensor/facility ratio, fewer but more selective graph neighbours improved the network survivability rate.

Moreover, the experiments showed that the closest-first greedy strategy outperformed all other strategies in terms of optimal recharge trips (i.e., initiated one-hop from recharging station). Even the single path proactive approach outperformed the passive solution in terms of cumulative number of losses until equilibrium. Finally, even though the passive approach reached a perfect balanced state (i.e. equilibrium without sensor losses), this was achieved using twice the number of recharging sockets when compared to the proactive approach. In general, our simulation results exposed several trade-offs between the key variables (i.e., topology, transmission range, locomotion cost, sensor knowledge and station role) and their impact on network survivability and continuous coverage.

1.3 State of the Art

To extend the lifetime of sensor networks, researchers have attempted to obtain alternative sources of energy to power their sensors. So far, the search had been limited to extract or "harvest" energy from the environment in various forms. The most common approach is the use of solar panels as well as vibrations (e.g., [33, 58]). Recent advances in wireless technology have made possible the idea of recharging wireless devices using electromagnetic induction or resonance of electromagnetic waves. An example can be found in [26], where power is transmitted short distances to recharge bio-implanted microsystems. In this case, the electromagnetic transfer occurs between an external coil and an internal receiving coil. This approach allows patients with biomedical implants to recharge the batteries periodically by using a wireless transmitter and thus avoiding unnecessary surgeries. Other examples of wireless recharging can be found in (e.g., [2, 3, 49]).

A closer look at the existing literature on energy management strategies for sensor networks reveals two main categories of solutions: cluster-based approaches and mobility-based approaches.

1.3.1 Cluster-based Energy Management

Examples of cluster-based strategies applied to wireless sensor networks can be found in [28, 45, 57, 83] with LEACH and HEED being some of the most widely studied. In particular, Rahimi et al. [58] introduce the concept of energy harvesting or energy distribution by providing sensor nodes (i.e., robots) with solar panels. These robots become energy producers while the rest of the sensors act as energy consumers. The area is divided into energy cells. The number of robots per cell depends on the number of static nodes, the amount of energy that robots consume when they move and

the amount of energy that can be stored in static nodes. The robots send periodical requests to all sensors using a flooding technique and sensors in need of energy will respond with their location and estimated life expectancy. The robots then move to service the nodes with the most urgent need for energy.

In environments where the access is limited due to hazardous situations, the deployment of mobile autonomous robots is a possible alternative (e.g., [35, 38, 48]). In particular, [48] considers three scenarios for repairing a sensor network using mobile robots: 1) Centralized manager, 2) Fixed distributed manager and 3) Dynamic distributed manager. In the first two cases each robot is assigned to an equally sized area. In the first case, one central manager receives all the requests from all areas and dispatches the corresponding robot to service it. In the second approach, each robot has double functionality, receiving all requests from its area and responding to them using a FIFO approach. Finally, in the third scenario, the area is dynamically divided in sub-areas containing exactly one robot. Sensors select the closest robot as their maintenance robot. This structure represents a Voronoi diagram with robots as cluster heads. The robots receive all failure requests from their region and proceed to replace the faulting node. As robots move, they broadcast their position to all sensor nodes within the region. The robot's location is also relayed to other sensors in the neighboring regions to update their membership. As robots move to replace faulting nodes, the composition of the Voronoi partitions changes accordingly.

Sheng et al. also explore the use of service robots for maintenance tasks in active sensor networks (e.g., [64, 75]). In particular, they address the issue of robot workload distribution and task allocation. Their algorithms assume that sensors are already localized through a distributed localization algorithm. Then, starting with a random unbalanced Voronoi partition, the robots exchange information with adjacent regions

and update their sensor lists until a balanced partition is achieved. In [75] Sheng et al. examine the problem where service robots have to visit a number of points of interest for network management related tasks. Their first algorithm deals with the itinerary of robots within their Voronoi partitions and the second part deals with the construction of a balanced Voronoi partition. In this case, the criterion for load balancing is the combination of two metrics: the traveling distance of the robots and the cost of servicing each location. Similar to [64], the robots start with a random partition and, in each iteration of the algorithm, they exchange load information with the adjacent robots, recalculating the Voronoi partition until a balanced distribution among all robots is obtained. The emphasis of the experimental analysis of their algorithms centers on the quality of the partition rather than the cost of achieving such partition. Sheng at al. observed that due to the discrete nature of the point of interest, in this case sensors, it was very difficult to obtain a perfectly balanced partition. Consequently, the output of the algorithms may lead to oscillatory behavior. To avoid this behavior and achieve convergence, the conditions for termination are relaxed to check for changes to the standard deviation of the load distribution and stop when no more improvements are seen after a predefined number of iterations.

The analysis of the existing cluster-based energy management approaches reveals some limitations, such as:

1) Scalability: solutions based on predetermined partitions of the field (i.e., geometric shapes) or centralized approaches do not perform well as the size of the network increases and may lead to an unbalanced workload distribution.

2) High communication cost: In partitions where cluster membership is dynamic, there is a significant overhead since nodes need to be notified of any change in the position of their cluster heads (i.e., maintenance entities).

3) Unbalanced workload among maintenance entities: If the distance to the maintenance robots is the only criterion for clustering, the partition can become very unbalanced due to the robot movements in the field.

1.3.2 Mobility-based Approaches to Energy Management

The idea of adding mobility to specific network elements has created a new stream of WSN known as Mobile Sensor Networks (MSN). Adding mobility to the sensors has expanded the scope of their applications, from terrestrial, to aerial, to marine deployments (e.g., see Figure 1.1).

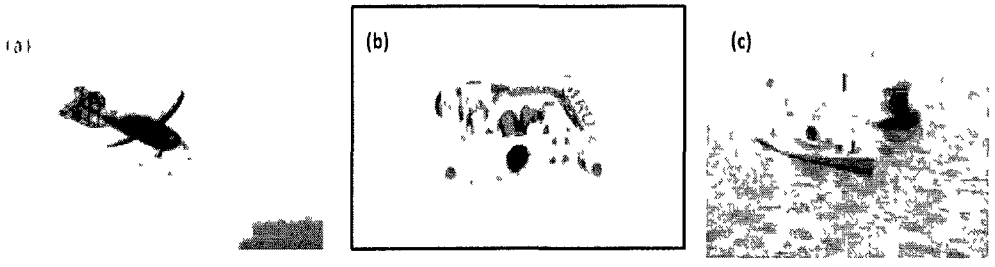


Figure 1.1: Mobile sensors. a) MMALV-aerial/terrestrial, b) Mini-Whegs [7], c) NAMOS [70]

For some MSN deployments, the focus has been moved to the base stations as a mechanism to balance the energy levels among all sensors. For example, it is observed in [46], that sensors closer to the base station tend to deplete their batteries much faster than other sensors. These sensors have to route/aggregate data flowing from remote parts of the network towards the base station. This disparity creates bottlenecks in areas closer to the base station. To counteract this effect, the authors proposed the use of mobile base stations and studied different mobility strategies. The authors found that for circular deployments, routes that followed the periphery of the circle, combined with short path routing had the best overall performance.

When the responsibility for maintaining a satisfactory level of energy falls to the sensors and not to the mobile recharging stations, the standard method of deciding when to recharge has been based on fixed thresholds [80]. In this case, the sensors are able to compute their remaining operational time and coordinate the use of the service stations [49]. Furthermore, for instances where the sensors or robots have to visit a predefined number of point of interests, [80] describes threshold vs. non threshold-based solutions in which robots decide to visit service stations depending on their proximity and the nature of the points of interests.

In general, mobility-based solutions to energy management attempt to extend the network lifetime by reorganizing the network components and thus overcoming the disparity in terms of energy degradation. The use of mobile relaying sensors [79] or mobile base stations will help to increase the network operating life, but the loss of coverage due to battery depletion will be inevitable since there are no provisions to recharge or replace sensors in a sustainable manner.

A more detailed review of previous works is presented throughout the thesis when examining the proposed models.

1.4 Methodology and Thesis Outline

The methodology used in this research adheres to the following general guidelines:

1) The problem of sensor network maintenance is divided into two broad categories: networks of static sensors with mobile facilities and networks of mobile sensors with static facilities. It is important to distinguish between the concepts of service facilities or recharge stations and the concept of base stations. Service/recharge facilities are not base stations and their functionality is limited only to network repair and maintenance tasks.

2) For each of the main categories, the research focuses on finding solutions which extend the network lifetime by replacing or recharging depleted sensors. In particular for static networks, parameters such as workload balance, number of messages, network size, distance to the service facility and total facility travel time are taken into account in the proposed models. For mobile networks, parameters such as topology, number of local neighbours, number of recharging sockets, transmission range and mobility cost are considered in the design.

3) The analytical properties of the proposed algorithms are examined with emphasis on correctness, convergence and/or termination.

4) The proposed solutions are implemented and validated through a series of prototypes using Omnet++. This simulator provides a stable and reliable environment for wireless ad-hoc and sensor networks with a vast number of supported protocols including IEEE 802.11. Omnet++ offers a continuous development as well as an active community of developers, researchers and conferences. The experiments are designed to compare the performance of the proposed solutions and test the impact of several variables such as: network size, number of messages, quality of the solutions, cumulative number of failures over time and time needed to reach energy equilibrium.

The structure of the thesis is as follows: In Chapter 2 we describe the static network scenario and the cluster based energy management solution. The extension of the balanced cluster solution to solve the Facility Location Problem is presented in Chapter 3. For both solutions, experimental results are also discussed. In Chapter 4 we examine the problem of network maintenance in a mobile sensor scenario as a particular instance of the Frugal Feeding Problem (FFP). The applicability of a passive vs. a pro-active approach is discussed along with the corresponding experimental analysis. In Chapter 5, the proactive approach to the FFP is enhanced to study the impact of several key variables (e.g., topology, transmission range, locomotion and sensor knowledge). Finally, in Chapter 6, we present some conclusions and propose the next steps to enhance the research.

Chapter 2

Strategies for Static Sensor Networks

2.1 Introduction

Existing solutions for energy management (with or without robots) rely on some kind of clustering or partitioning of the network and the use of maintenance entities as cluster heads. The energy management problem is basically addressed by 1) either creating a fixed partition of the field [48], which limits the scalability of the solutions, or 2) constructing and maintaining dynamic clustering structures that depend on the current position of the cluster heads (e.g., [48, 58]). The latter approach, where cluster membership is dynamic, incurs a significant overhead since nodes need to be notified of any change in the position of their cluster heads. When robots are employed in the solution, they act as the cluster heads; in this case, it is possible for the cluster membership to become very unbalanced due to robots' movements in the field. A significantly unbalanced partition will impose extra burden on the robots' resources compromising the health of the entire system. For these reasons, we focus on energy management with a static cluster membership where each robot acts as a cluster head initially positioned at the center of mass of its cluster.

Assuming the recharging or replacing capacity of each robot is bounded and the same among all robots, the problem lies in finding a network partition where the robots act as a cluster head, and: (1) each robot minimizes the sum of the distances to all sensors in the cluster, and at the same time (2) the number of sensors in the clusters is balanced. This problem should be solved efficiently, at least for the weakest

elements of the system (i.e., the sensors); that is (3) the number of sensor communications should be kept small. Finding an optimal placement for the maintenance robots is a NP-hard problem [62].

In this section we propose a balanced cluster-based energy management system to recharge or repair a network of static sensors by employing mobile robots. Our approach combines a static cluster partitioning with a distance aware robot positioning. Based on existing experiences with clustering techniques (e.g., [4, 48]), our objective is to minimize the number of messages exchanged by sensors as well as the total robot travel time.

Our balanced maintenance system (BMS) creates clusters of sensors where each mobile robot becomes a cluster head. The clusters are created through a sequence of iterations, each creating a partition whose quality is strictly better than the previous. In particular, the cluster membership in each iteration will change only if the quality of the solution improves by accepting a new sensor. Convergence (i.e., no further improvement to the clustering partition) is achieved in a finite number of steps; when that happens, cluster membership becomes static and robots reposition themselves at the center of mass of its cluster.

The main differences between the work of [64, 75] and ours are:
1) The problem we attack is a *bi-criterion optimization* one: the partitioning of the sensors and the positioning of the robots within their cluster must be such that the service movement of the robots in their cluster is minimized *and* at the same time the workload of the robots must be optimized (i.e., the size of the clusters must be the same); in [64, 75] only one (i.e., the first) optimization criterion is used. Once convergence is achieved, the number of sensors in each cluster is balanced so that, in

general, the robots maintain the same number of sensors.

2) We propose using a static partition as opposed to dynamic structures that need to be updated when the robots move to repair or service a node.

3) We have robots re-position themselves within their regions to minimize the total trajectory required to recharge or repair nodes in the cluster.

4) We require low sensor communication. Once the balanced clusters are created, membership becomes static and there is no need inform the sensors about new cluster heads.

2.2 The Model

The proposed cluster-based energy management approach can be used to either replace or recharge the sensor nodes with minor modifications to the algorithms. Regardless of the kind of functionality selected, the model includes the following key components:

1) A set of N sensors, $S = \{s_1, ..., s_N\}$, randomly distributed in an area of unspecified shape.

2) A set of K robots, $\{r_1, ..., r_K\}$, also randomly distributed throughout the area.

3) Robots can move anywhere within the area and they all move at the same speed.

4) Sensors are static and can communicate with other sensors and robots within their transmission range (R).

5) Robots can communicate with sensors and robots within their transmission range (R').

6) Sensors are be able to monitor their energy levels and compute remaining time or time to total depletion (T_d).

Following a widely accepted practice in modeling and simulation of WSN (e.g., [20, 50, 60]), all the interactions among sensors and robots assume the existence of an ideal MAC Layer. An ideal MAC layer provides a reliable wireless communication channel by guaranteeing a collision-free access to the medium and eliminating interference produced by receiving simultaneous radio transmissions.

The proposed clustering algorithms assume that robots and sensors can determine their own positions by using GPS or other localization methods. However, there is no requirement for a global clock or centralized entity to coordinate communications or actions; that is, the system is asynchronous. It is assumed that sensors and robots

are deployed creating a single-hop network similar to (e.g. [28, 83, 86]) where the sensors are able to adjust their transmission ranges (R) to reach any other sensor or robot in the network if necessary. This assumption will be relaxed in Section 6.2.1 when we examine the multi-hop network scenario.

Sensors and robots are grouped in clusters creating a balanced partition of the entire area. A partition is considered *stable*, and therefore final, when there are no changes in the cluster membership. The stable partition must then satisfy the load balance requirement based on predefined metrics. For example, the number of sensors in the clusters combined with their energy levels, etc. Without loss of generality, the model will use a simple metric based on the number of sensors in the cluster (i.e., equi-partition). Consequently, there will be $NmodK$ clusters containing exactly $\lceil N/K \rceil$ sensors and the rest will contain $\lfloor N/K \rfloor$ sensors.

2.3 The System

Our balanced maintenance system creates a cluster structure that facilitates the network maintenance tasks needed to maximize the network lifetime. This is a two stage system. The first phase, the initialization phase, creates a stable and balanced partition and the second phase is where the maintenance tasks are carried out by the robots.

Once the clustering creation is finalized, there will be exactly one maintenance robot for each sensor in S, the sensors will know the location of their maintenance robot, and each robot will know the position of all the sensors in its cluster. The sensors will also keep the location of two other robots (i.e., peers) at most for failure detection. Once a stable and optimal partition has been achieved, the robots will

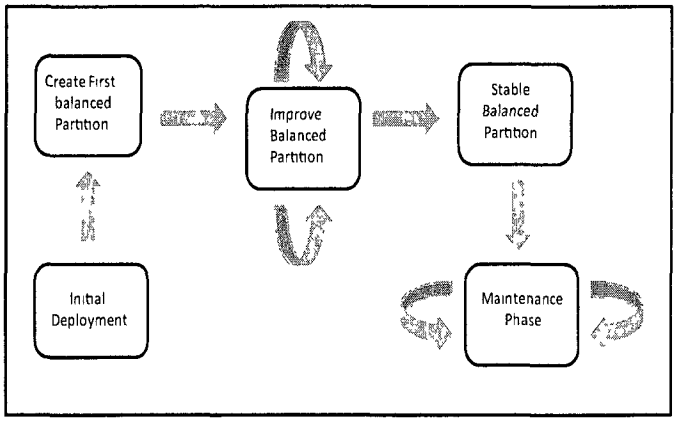


Figure 2.1: System life cycle

compute the sensor pairing and notify each sensor about its peers. At this point, the new location of the robot can also be sent to all cluster members if necessary.

After the initialization phase has been completed, the robots and sensors move to the maintenance phase. The behaviour for sensors in this phase is rather simple and it basically includes the following actions:

1- Sensors monitor their energy levels. When a sensor detects that its time-to-depletion or time left (T_l) is less than a constant (T_d) then it sends a request for recharge (RC) to its maintenance robot.

2- Sensors send probes periodically to their peers. If a response is not received; they send a repair request (RR) to their maintenance robot containing the location of the faulting node.

Robots, on the other hand, have a more active role in the maintenance phase. At the beginning of this phase, the robot is positioned at the center C of the cluster. Then, it awaits repair/recharge requests from its sensors and tries to service them in a first-come first-serve manner whenever possible. Figure 2.1 summarizes the life cycle of the proposed maintenance system.

The maintenance phase relies heavily on the cluster partitioning obtained during the initialization phase. The quality of the partition, as well as the resources needed to achieve convergence, have a significant impact on the health of the overall system. Therefore, the emphasis of the algorithm design and experimental analysis centers on how to achieve this cluster partition in a distributed and cost effective manner.

2.4 Computing the Stable Partition

The initial phase of the cluster creation begins by having the robots broadcast their locations to all sensors; the sensors will respond to only one robot (e.g., the closest) by sending a CLUSTER_JOIN request with its location. The goal of this phase is to obtain a Voronoi partition of the field with K clusters containing exactly one robot as cluster head.

At this point, all sensors know about their tentative maintenance robots and all robots know the location of the sensors they have to maintain. Then, the robots should move to a point in their clusters that minimizes the sum of all distances. Such a point is unique and is known as the Weber point. Unfortunately, the Weber point is not computable for $N \geq 5$ sensors [8, 13]. Consequently, the robots will use the center of mass as an approximation of the Weber point. The center of mass of a set of N points (x_1,...,x_N) with masses (m_1,...,m_N) is defined as the point C, which satisfies that: $C = \frac{1}{M} \sum x_i m_i$ where $M = \sum m_i$ with $1 \leq i \leq N$. Each robot computes the center of mass for the sensors in its cluster and moves towards that location. In our scenario it can be assumed, without loss of generality, that the mass of each point (i.e., position of sensors) is equal to 1. Consequently, the robots will relocate to the point C in their Voronoi region that satisfies: $C(c_1, c_2) = (\frac{1}{N} \sum x_i, \frac{1}{N} \sum y_i)$ where

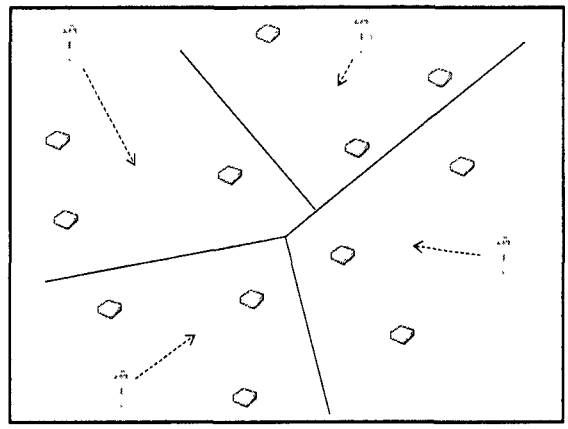


Figure 2.2: Cluster creation. The robots move toward the center of their regions.

(x_1,y_1),...,(x_N,y_N) are the positions of the N sensors in the region. See Figure 2.2 for cluster formation and repositioning.

Once the robots have arrived at their cluster centers, they will repeat the same process until there are no more changes to their cluster membership.

2.4.1 Computing the Stable Partition with Limited Memory

A simple distributed algorithm with "limited memory" can be used for the initialization phase. A robot is said to have "limited memory" if it only has capacity to store (i.e., remember) information about the previously obtained cluster. In this algorithm the robots operate in logical rounds or iterations, sending START_ROUND broadcast messages to all sensors, announcing their initial positions. After receiving the initial position from all robots, the sensors will rank the robots based on their distances (e.g., from closest to farthest) and will reply with a JOIN_CLUSTER message to the smallest ranked robot.

The robots will accept the first N/K requests and reject all others by sending

a CLUSTER_FULL message back to the sensor. A sensor that receives a CLUSTER_FULL message will then send a CLUSTER_JOIN message to its next ranked robot (i.e., second closest, etc.). After accepting its sensors, each robot sends an END_ROUND message to the other robots; proceeds to compute the center of the newly obtained cluster and travels to that location. The robots can only "remember" the previously created cluster. Therefore, the algorithm terminates when two consecutive rounds produce the same clustering structure, and at that point the robots will send an INIT_DONE message. An obvious limitation of this method is that, due to the asynchronous and distributed nature of the network, there is no guarantee that the algorithm will always converge to a stable partition.

Let us consider the example shown in Figure 2.3. Due to the asynchronous nature of the communications, the "limited memory" algorithm does not converge in this particular environment.

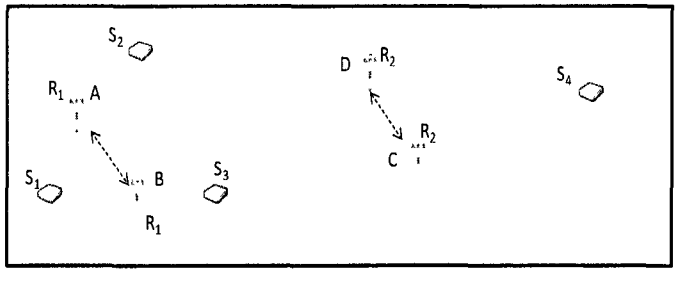


Figure 2.3: Simple limited memory algorithm

Let us assume that during the first iteration of the algorithm, robot R_1 receives a sequence of CLUSTER_JOIN messages from sensors S_1, S_2, S_3, S_4 in that order. According to the algorithm, robot R_1 will accept sensors S_1 and S_2 and send CLUSTER_FULL messages to sensors S_3 and S_4. After being rejected by robot R_1, sensors S_3 and S_4 will send CLUSTER_JOIN messages to their robot with ranking 2, in this case, R_2. Once this phase has been completed, robot R_1 moves to point A and robot

R_2 moves to point C. Let us assume now that for the second iteration, after the corresponding broadcast messages have been sent by both robots, robot R_1 receives a new sequence of CLUSTER_JOIN messages from sensors S_1, S_3, S_2 in that order. Following the same behaviour, robot R_1 will now accept sensors S_1 and S_3 and robot R_2 will eventually accept sensors S_2 and S_4. The robots will then move to point B and D respectively. Since a new partition has been created, the robots will proceed to a new iteration.

Consequently, if consecutive iterations of the algorithm show a changing behaviour regarding the arrival of CLUSTER_JOIN messages as described in the two iterations above, Robot R_1 will be oscillating between points A and B and robot R_2 will oscillate between points C and D without achieving a stable partition. The previous example could be generalized to state that if the robots are provided with a fixed amount of memory $O(M)$ (i.e., the robots can only "remember" the M previously created clusters), the "simple memory" algorithm does not always converge.

2.4.2 Achieving Convergence

In order to achieve convergence of the clustering structure, we propose to extend the memory capabilities of the robots. In particular, the robots will now be able to "remember" all previous clusters they have created and will stop the execution of the algorithm as soon as they obtain a previously seen or created cluster. The behaviour of the static sensors remains unchanged in this new approach and there are only small changes to the robot's behaviour. Each robot will now keep a list of old clusters and will stop the execution when it finds a newly created cluster on such a list. The modified behaviour for the END_ROUND state is summarized by the Algorithm 1.

Algorithm 1 Initialization with Memory: Robot r

```
(* In State END_ROUND : *)
if receiving END_ROUND(r') then
    numOfEndRoundMsg = numOfEndRoundMsg+1
    if numOfEndRoundMsg = K-1 then
        C = center(currentCluster)
        if find(currentCluster,clusterList) then
            send INIT_DONE broadcast message
            become DONE
        else
            MoveTo(C)
            become INIT
        end if
    end if
end if
```

Lemma 1. *The cluster formation with the memory option will converge in a finite number of rounds.*

Proof. Let $S = \{s_1, ..., s_N\}$ a set of N static sensors randomly placed in a plane and $\{r_1, ..., r_K\}$ a set of K mobile robots also randomly placed in a plane. At any given round of the clustering algorithm, a partition P containing exactly K clusters $\{C_1, ..., C_K\}$ is obtained.

Without loss of generality, we can assume that at the end of the first round the algorithm produces the following K clusters $C_1, C_2, ..., C_K$ where $C_i = \left\{r_i, s_{\frac{N}{K}i+1}, s_{\frac{N}{K}i+2}, ..., s_{\frac{N}{K}(i+1)}\right\}$ with $0 \leq i \leq K-1$. Let $P = \{P_1, ..., P_f\}$ be the set of partitions created after each round of the algorithm, where $P_i = \{C_1^i, ..., C_K^i\}$ is the cluster partition obtained after round i and C_j^i with $1 \leq j \leq K$, are the clusters obtained in such round. By definition of the clustering with memory algorithm, P_f is the final partition where no further improvement is achieved. This means that $\forall C_i^f$ $(1 \leq i \leq K)$, $\exists j$ $(1 \leq j < f)$ such that $C_i^f = C_i^j$. The existence of the final partition P_f can be proved by contradiction.

Let us assume that the algorithm does not converge. This means that after each round, at least two robots have accepted or rejected new sensors. Consequently, a new partition is created where there is at least one cluster C_i^f, $1 \leq i \leq K$, such that $\forall j$, $1 \leq j < f$, $C_i^f \neq C_i^j$. This contradicts the fact that there is only a finite number of combinations of N/K sensors for the same cluster head r_i. Hence, the Lemma holds. □

2.4.3 Improving Convergence and Quality - Sensor Selection

In the two previous algorithms, the robots limit themselves to only accepting the first N/K sensors and blindly rejecting any other request. This rather passive behaviour, in combination with the asynchronous nature of the network, may impact the performance of the algorithm.

In an effort to improve convergence and the quality of the overall partition, a new sensor selection rule is added to the iterations. The modified algorithm will still execute in rounds, where each round is determined by START_ROUND and END_ROUND messages sent by the robots to coordinate their movements. The difference now is that, after accepting the first $\frac{N}{K}$ CLUSTER_JOIN(s,r_{next}) requests, where s represents the sensor interested in joining the cluster and r_{next} is the next ranked robot in s list, a robot will still accept a new sensor in its cluster as long as the inclusion of the new sensor decreases the diameter (i.e., measured as the Euclidean distance) of the cluster.

Since our model imposes the requirement of load balancing, the robots will then have to reject one of the previously accepted sensors in order to accept a better candidate. The selection of which sensor will be excluded from the cluster depends on the sensor's distance to the center and its proximity to a neighboring robot which

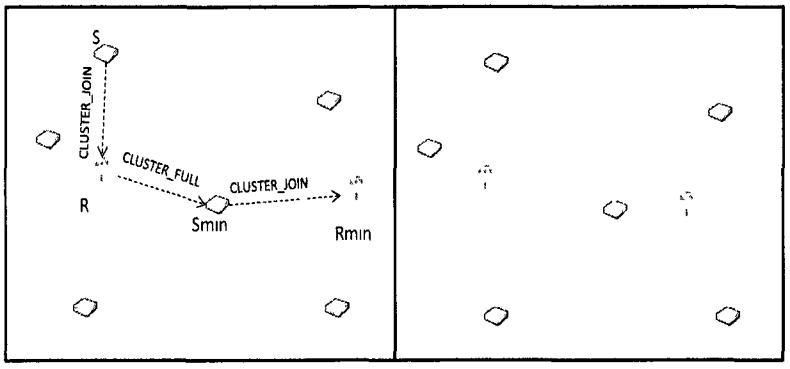


Figure 2.4: Sensor Selection.

has not previously rejected that sensor (i.e., its next ranked robot). Basically, this rule can be called the "farthest and closest" rule where the sensor farthest from the center and closest to a neighboring robot will be rejected. Ties are broken based on a secondary criterion, such as energy levels or simply selecting the sensor with the smaller Id (see Figure 2.4).

Rule 1. *Farthest and Closest Sensor*

Let $\{r_1, ..., r_K\}$ be a set of K robots with clusters $\{C_1, ..., C_K\}$. If a robot r with cluster C_r containing the sensors $\left\{s_1, ..., s_{\frac{N}{K}}\right\}$ receives a CLUSTER_JOIN message from sensor s, then r accepts s if and only if $\exists\ s_{min} \in C_r$ and a robot $r_{min} \neq r$, such that $d(r, s_{min}) > d(r, s)$ and $d(s_{min}, r_{min}) = min\{d(s_i, r_j)\}$, where $s_i \in C_r$, r_j is the next closest ranked robot in s_i list, and d denotes the Euclidean distance.

The modified behaviour for a robot which receives a CLUSTER_JOIN message after completing its cluster is presented in Algorithm 2.

Algorithm 2 Sensor Selection: Robot r

```
(* In State END_ROUND : *)
begin
if receiving CLUSTER_JOIN(s, r_next) then
    s' = FindFarthestandClosest(currentCluster,s)
    if s'=null then
        send CLUSTER_FULL message to s
    else
        send CLUSTER_FULL message to s'
    end if
end if
if receiving END_ROUND(r') then
    numOfEndRoundMsg =numEndRoundMsg+1
    if numOfEndRoundMsg = K-1 then
        C = center(currentCluster)
        MoveTo(C)
        if find (currentCluster,ClusterList) then
            send INIT_DONE broadcast message
            become DONE
        else
            become INIT
        end if
    end if
end if
end
```

The benefits of using the sensor selection algorithm vs. the "memory only" option become clear by examining the network depicted in Figure 2.5. However, there is an extra cost in communication since new CLUSTER_FULL messages are generated every time a node is rejected and this rejection triggers a new CLUSTER_JOIN message that is sent by the sensor in an attempt to join another cluster. This presumed extra cost should be canceled by the fact that the sensor selection algorithm needs fewer iterations to achieve a stable partition (i.e., convergence point) than the "memory only". The experimental results shown in Section 2.5 will confirm this assertion.

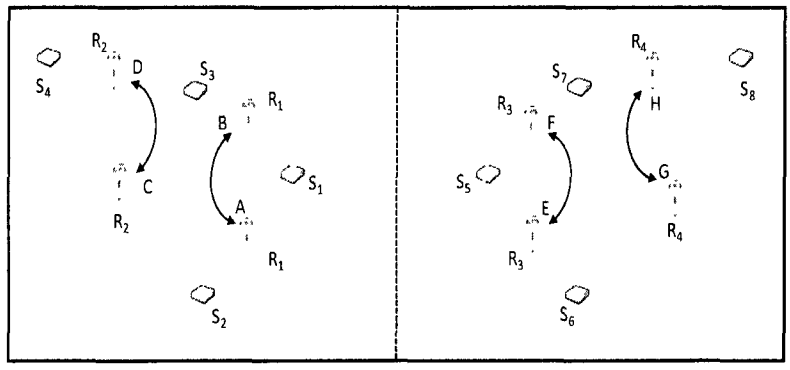


Figure 2.5: Memory vs. Sensor Selection algorithms in a network with 8 sensors and 4 robots.

For simplicity, the network in Figure 2.5 is divided in two parts where the right hand side mirrors the left one. Due to the asynchronous nature of this network, we can assume that during the execution of the algorithm both sides are slightly out of sync. This means that the movements of the robots and sensors' replies on the right side do not necessarily mirror those of the left side.

For instance, let us assume that after one round, robots R_1 and R_2 move to points A and D and robots R_3 and R_4 move to the points F and G respectively. If in the subsequent rounds either side experiences this behaviour, where robots alternate

between points but out of phase with respect to the other side of the network, the application of the "memory only" algorithm will converge after 4 rounds whereas the sensor selection algorithm will converge after only 2 rounds.

Lemma 2. *The cluster formation with sensor selection terminates after a finite number of rounds.*

Proof. Let $S = \{s_1, ..., s_N\}$ a set of N sensors and $\{r_1, ..., r_K\}$ a set of K mobile robots. At any given round of the clustering algorithm, robot r_i, with $1 \leq i \leq K$ receives L CLUSTER_JOIN(s_j, r_{s_j}) requests from sensors $s_1, ..., s_L$ where $1 \leq \frac{N}{K} < L \leq N$, $1 \leq j \leq L$ and r_{s_j} is the next closest robot to sensor s_j.

After accepting the first $\frac{N}{K}$ requests, robot r_i can still accept the next $L - \frac{N}{K}$ sensors if $\forall s_l$, with $\frac{N}{K} < l \leq L$, s_l satisfies the *farthest and closest* rule (see Rule 1). For simplicity, we can assume that if L is known, robot r_i will eventually accept the first $\frac{N}{K}$ requests in the ordered set $\{(s_1, r_{s_1}), ..., (s_L, r_{s_L})\}$ where $d(s_j, r_{s_j}) < d(s_{j+1}, r_{s_{j+1}})$ and $d(s_j, r_i) < d(s_{j+1}, r_i)$. From Lemma 1, we know that for the non-ordered case, the algorithm converges if the robots have memory capabilities. Hence, this Lemma also holds.

□

2.4.4 Properties of the Stable Partition Computation

In the two previous sections we have presented two techniques to achieve a clustering partition in a distributed manner. The stable and balanced partition can be achieved if the sensors have memory capabilities and convergence can be improved by using the sensor selection algorithm. This section discusses some of the main properties of the clustering algorithms and the final stable partition.

Proposition 1. *The stable partition contains exactly K clusters, where K is the number of robots, and each sensor is assigned to only one cluster.*

Proof. By contradiction, assume that the proposition does not hold. This means that the stable partition either contains less than K clusters or that a sensor was assigned to more than one cluster. If the final partition contains less than K clusters, where K is the number of robots, then $\exists r_i \in \{r_1, ..., r_K\}$, that did not receive any CLUSTER_JOIN requests in any of the iterations of the memory only or sensor selection algorithms. This contradicts the fact that, in each round, every robot $r_j, i \neq j$, accepts $\frac{N}{K}$ CLUSTER_JOIN requests, and rejects all others by sending CLUSTER_FULL messages. Therefore, there will be exactly $\frac{N}{K}$ sensors that received CLUSTER_FULL messages and will be available to join r_i's cluster.
Furthermore, for a sensor to be assigned to more than one cluster, it has to be accepted by two robots simultaneously (i.e., in the same round). This contradicts the fact that in each round, there will be only one active CLUSTER_JOIN message per sensor. A sensor will not attempt to join another cluster (i.e., by sending another CLUSTER_JOIN message) unless it has been previously rejected, which makes it impossible for a sensor to be accepted by two simultaneous robots.

□

Proposition 2. *The stable partition produced by the sensor selection algorithm satisfies the condition of load balance and all sensors are assigned to the closest possible robot.*

Proof. By contradiction, assume that in the last iteration of the sensor selection algorithm, the partition is not balanced. This means that $\exists r_i \in \{r_1, ..., r_K\}$, that received L CLUSTER_JOIN requests where $L > \frac{N}{K}$ and another robot $r_j, j \neq i$, that received L' CLUSTER_JOIN where $L' < L$. According to the algorithm, robot r_i will accept the closest $\frac{N}{K}$ sensors, reject the remaining $L - \frac{N}{K}$, and initiate a new iteration.

This means that the current partition was not stable and therefore cannot be final.

□

From Propositions 1 and 2 (i.e., correctness properties) and the Lemmas 1 and 2 (i.e., termination property), it follows that:

Theorem 1. *At the end of the initialization phase, a stable and balanced partition is obtained where every sensor is assigned to exactly one maintenance robot and the robots are placed at the center of their corresponding clusters.*

2.5 Experimental Analysis

This section examines the simulation results for the algorithms presented in Sections 2.4.1 and 2.4.3. In all cases the simulation software utilized was Omnet++ [77] along with the mobility framework extension [19]. To improve the robots' capabilities and provide additional control over their movements, a new mobility mode was implemented and added to the mobility framework. This new mobility mode inherits its basic behaviour from the constant speed mobility module but adds a new reactive approach where the robots will move on-demand after receiving a message. The destination will be the computed center of mass based on the senders' original positions. For all the experiments, the robots and sensors are randomly placed in an area of 1000x1000m^2. Sensors are static and once placed cannot be relocated but robots are mobile and there is no restriction on their movements.

The analysis centers on two important aspects of the solutions: communication cost and the quality of the partition. The tests are designed to measure the cost in terms of sensor communications as well as measure the quality of the clustering partition after each iteration and its progression towards convergence.

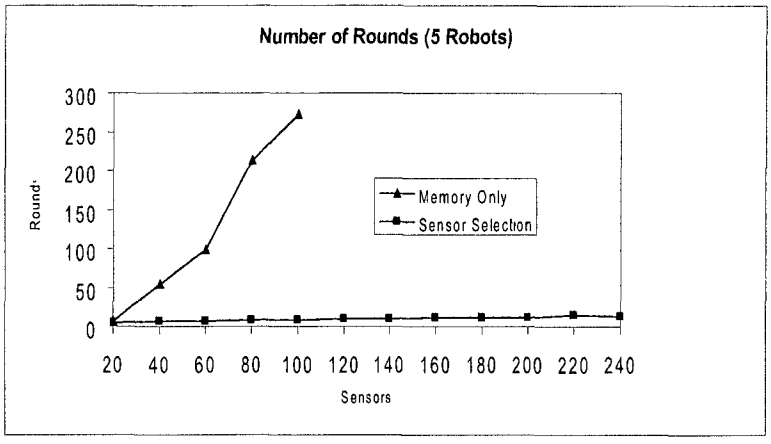


Figure 2.6: Number of rounds needed to converge in a network with 5 robots and 240 sensors

2.5.1 Rounds and Communication

The first experiment involves a performance comparison between the "memory only" and sensor selection algorithms in networks with 5 and 10 robots and up to 240 sensors. The number of rounds necessary to achieve convergence as well as the number of sensor communication required are plotted in figure 2.6 and 2.7 respectively.

The results of this test confirm what we discussed in Figure 2.5, that the sensor selection algorithm outperforms the "memory only" algorithm by a significant margin. In both cases, number of rounds and communication per sensor, the sensor selection algorithm shows an expected and almost constant communication cost per sensor even when the number of network elements increases.

The next natural step would be to examine the behaviour of the best algorithm (e.g., in terms of convergence) and take a closer look at the breakdown between sensor and robot communications under variable network size and number of robots.

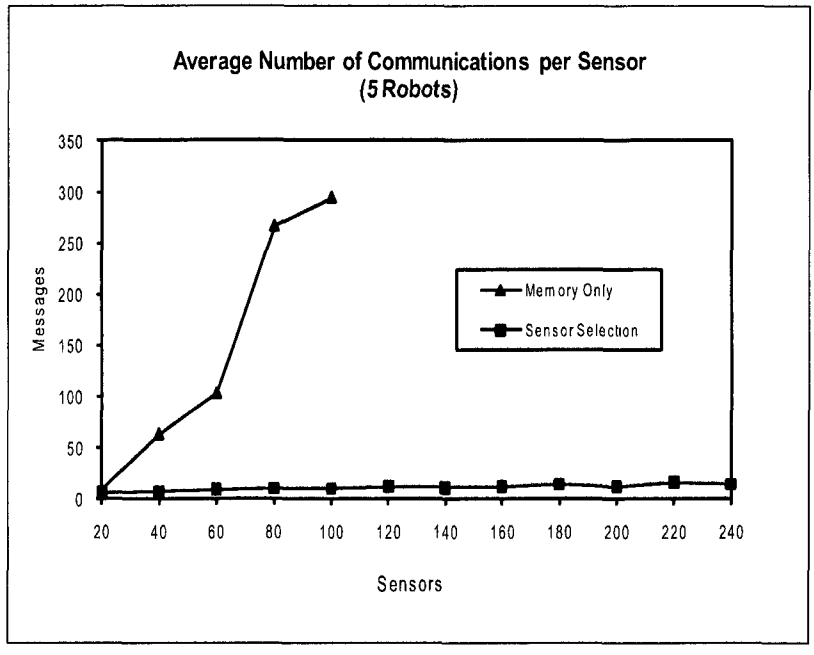


Figure 2.7: Communications per sensor in a network with 5 robots and 240 sensors.

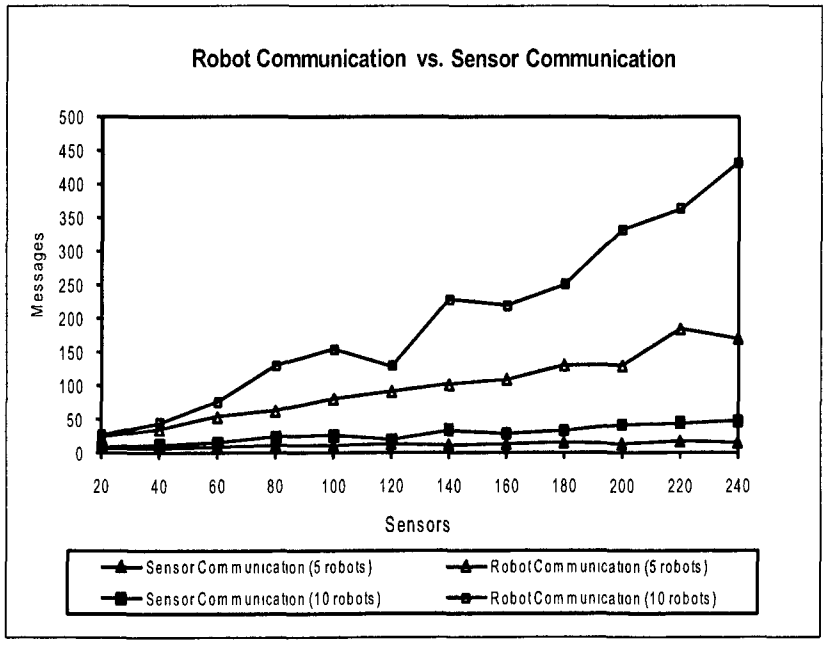


Figure 2.8: Robot and sensor communication cost for the sensor selection algorithm with variable network size.

Figure 2.8 clearly shows the difference of communication cost between robots and sensors. Robots have an active role in the clustering creation, limiting the sensor participation to a more passive role, which was one of our main goals. This behaviour frees the more resource-constrained sensors of the burden related to communication and coordination. It is also clear, by the results, that in networks with high sensor to robot ratio, the communication cost per sensor and robot is also lower than those with a larger number of robots. The larger the number of robots, the more frequently they will have to adjust their clusters by rejecting and accepting better candidates. This process generates a number of CLUSTER_FULL messages, which in turn will generate new CLUSTER_JOIN messages sent by sensors. This behaviour can be seen by the slight increase in sensor communication as the number of sensors increased and the number of robots is doubled from 5 to 10.

2.5.2 Quality of the Solution

This experiment explores the quality of the solutions to verify whether the fastest solution produces the best overall cluster partition. The quality criteria for this case are the average distance from a sensor to its cluster center (i.e., maintenance robot) and the average sum of the distances per cluster. The results of this test are compared with a centralized benchmark (e.g., K-Means). The K-Means is primarily used in resource constrained environments due to its low complexity and fast execution in large data sets and it has been widely used in wireless sensor networks (e.g., [16, 23, 30, 34, 40, 56, 74, 85]). According to [30], the general idea of the K-Mean clustering can be described as follows:

Consider an $l - dimensional$ space $D_1 \times \cdots \times D_l$. Let $d(p, q)$ be the distance between two points p and q. Given a set of n points $S = \{s_1, ..., s_n\}$ in the space

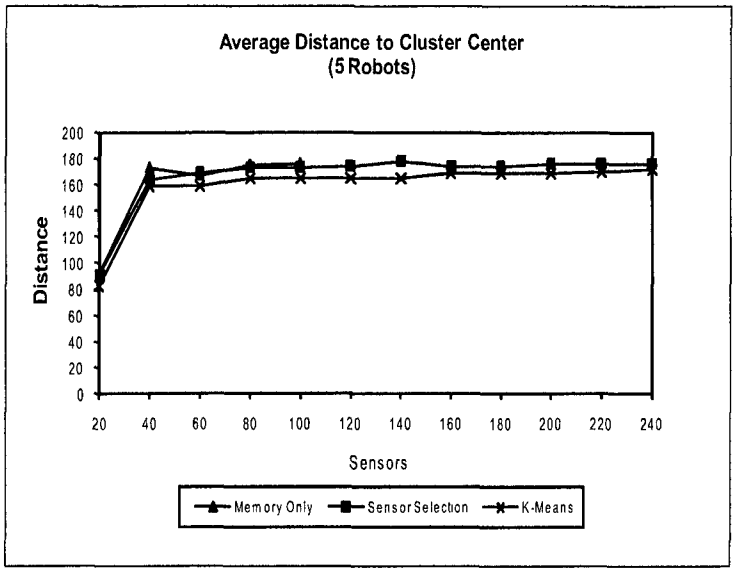


Figure 2.9: Average distance to cluster center comparison for networks of 5 robots and up to 240 sensors.

and a positive integer K, the K-Means problem is to find K points $C = \{c_1, ..., c_K\}$ (i.e, also known as centers), such that minimizes $\sum_{s_i \in S} min_{c_j \in C} d(s_i, c_j)^2$. The goal is to assign each point to its closest center while minimizing the sum of the distances between the points and their centers. The limitations of this method come from the fact that the number of clusters has to be set a priori and the quality of the final partition depends on the initial position of the centers.

Figures 2.9 and 2.10 show the comparison for networks of 5 and 10 robots with variable number of sensors. Contrary to the results for message costs, the three algorithms (including "memory only") produced very similar cluster partitions in terms of average sensor distance to their corresponding cluster centers. Another interesting finding is that the quality of the solution obtained by the algorithms did not change as we doubled the number of robots in the network.

Perhaps the most interesting finding is that the three solutions produced partitions

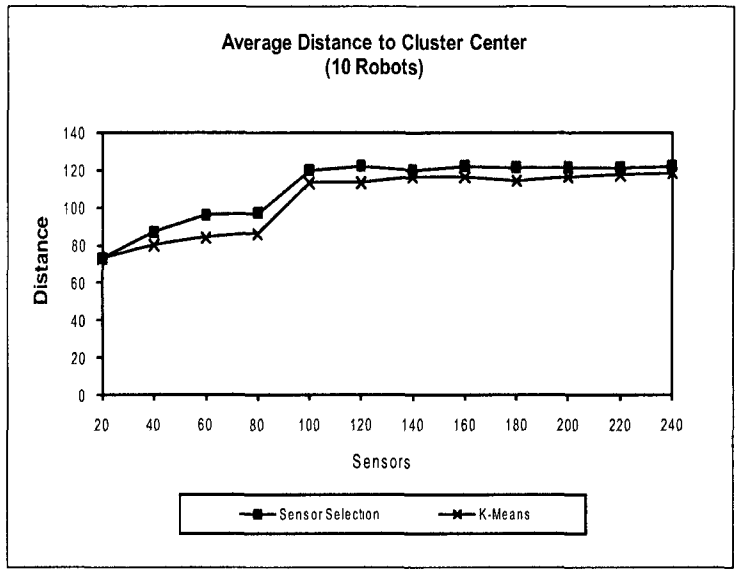


Figure 2.10: Average distance to cluster center comparison for networks of 10 robots and up to 240 sensors.

of similar quality when compared to the centralized clustering algorithm (i.e., K-Means). For this particular experiment, the same sensor deployment was used to execute a K-Means in SPSS for UNIX and the results were plotted along with our distributed solutions. For both quality parameters, average distance to cluster center and sum of distances, the experiment shows comparable results even when the K-Means partition did not contain the restriction of load balancing among all clusters. Figures 2.11 and 2.12 show the sum of distances for all clusters for all the solutions examined.

2.5.3 Progression Towards Convergence

The final test case examines the progression towards convergence and the quality of the partition after each iteration of the algorithms. For this test, several runs of the sensor selection algorithm were plotted. Figures 2.13 and 2.14 show the average distance to cluster center and sum of the distances respectively for each iteration of the algorithm until achieving convergence.

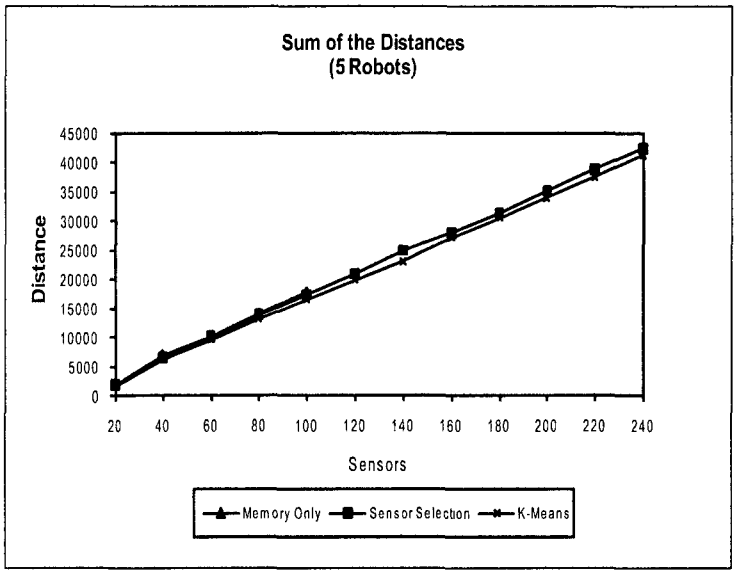


Figure 2.11: Sum of the distances in all clusters for networks with 5 robots and variable number of sensors.

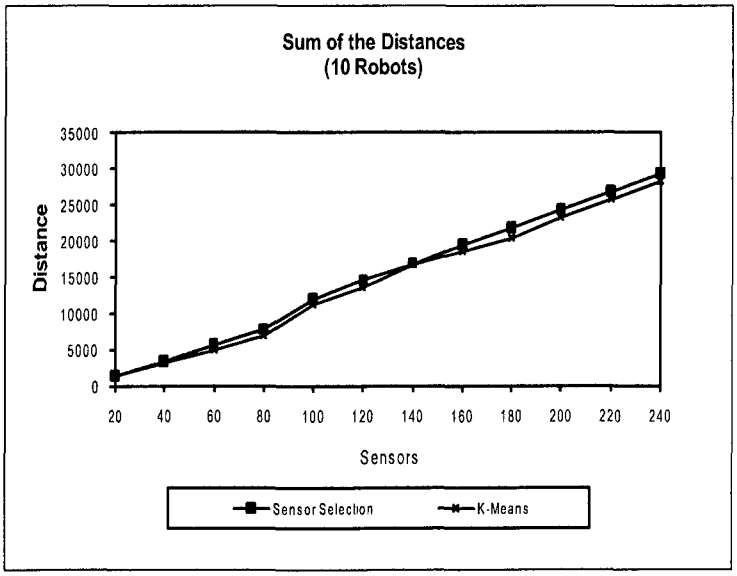


Figure 2.12: Sum of the distances in all clusters for networks with 10 robots and variable number of sensors.

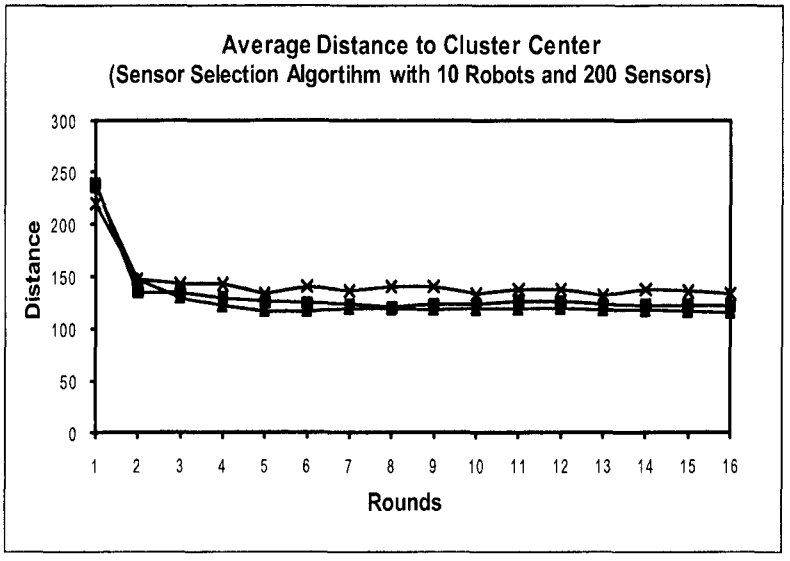


Figure 2.13: Average distance to cluster center after each iteration of the sensor selection algorithm.

The experiment shows a positive and somehow unexpected result, where progression towards convergence is not gradual as we may have anticipated. On the contrary, after only 3 iterations of the algorithm, the variations in quality are minor with an overall behaviour that is almost flat. This is an important feature of our approximation algorithms, which on average, if stopped after 3 iterations, will produce a solution close to the one achieved at the convergence point. Depending on the characteristics and requirement of the network, this behaviour could be an advantage in real life applications.

2.6 Summary of the Experimental Results

Throughout this section we have discussed the possibility of employing mobile robots to maintain or repair a network of static sensors. The final goal is to extend the operating life of the network by recharging or replacing depleted or faulty sensors with the help of mobile robots. The robots, on the other hand, should coordinate

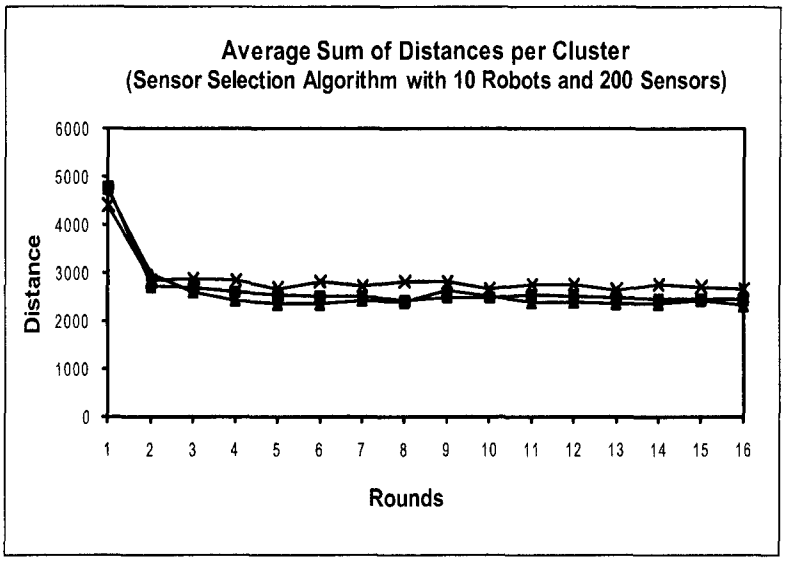


Figure 2.14: Sum of the distances after each iteration of the sensor selection algorithm.

their actions and balance their workload. This is achieved by partitioning the network into balanced clusters and positioning themselves at the center of these clusters. This work has also shown that the construction of such partitions in an asynchronous environment is possible if the robots are provided with some memory capabilities. Furthermore, with some simple modifications, a significant impact in cost reduction can be achieved by selecting and discarding the best candidate sensors in each round.

From the experimental analysis we can summarize the following general observations:

1. The sensor selection algorithm outperforms the simple "memory only" algorithm in terms of sensor communication cost.

2. Networks with higher sensor to robot ratio perform better than those with lower ratio in terms of average number of communications per sensors and robots.

3. Although very different in terms of communication cost, the "memory only" and

sensor selection algorithms produced a cluster partition with similar quality (i.e., similar average sensor distance to cluster centers and sum of the distances).

4. Both solutions produced partitions with approximately the same quality when compared to the centralized benchmark (i.e., K-means).

5. Fast convergence. The progression towards convergence of the clustering partition is not gradual, which translates into fast improvement in the initial 3 rounds and only minor adjustments afterwards.

The experimental results also show that the load balancing condition of each cluster containing exactly N/K sensors proves to be too restrictive in some situations. For certain sensor deployments, the final cluster topology, although balanced, is not optimal. Some clusters contain one or more sensors "relatively far" from their cluster peers. In an attempt to improve this situation, a new phase could be added to the aforementioned sensor selection algorithm. This post-processing or optimization phase involves the redistribution of some sensors between adjacent clusters as long as this operation improves the quality of the partition. The next chapter describes this approach and its application to the Facility Location Problem (FLP) in the sensor network domain.

Chapter 3

Distributed Facility Location for Sensor Networks

3.1 Introduction

In Chapter 2, we presented a mechanism to create a balanced clustering partition using static sensors and mobile maintenance robots. In this chapter we discuss how to improve an existing balanced partition based on predefined quality parameters and its application to a particular problem: the Facility Location Problem.

The Facility Location Problem (FLP), in its more general definition, attempts to find suitable locations for service facilities subject to a particular optimization criterion. For example, minimizing the client travel time to the facilities, which could be hospitals, schools, service stations, etc. In this chapter, we are interested in exploring the FLP in the sensor network domain, where mobile service facilities (e.g., robots) should be deployed in a way that guarantees a satisfactory level of service to all sensors in their areas, while achieving a workload balance among the facilities. In our case, the facilities will travel to the point of interests (i.e., sensors) to offer their service on-demand.

The FLP has been previously studied in the context of sensor networks (e.g., [24, 37]). However, the focus has been mainly on instances where a number of facilities are only placed in a subset of predefined locations. The goal in this case is to provide services to the sensors at minimal (traveling) cost. Finding an optimal placement is known to be a NP-hard problem [62]. Krivistki et al. [62] explore the

un-capacitated FLP in a sensor network environment. In particular, their problem is to find the optimal placement of K un-capacitated facilities (i.e., no restriction on the number of clients being served) out of M possible locations. Their solution starts with an initial deployment and attempts to either: 1) move a single facility to an available location, 2) close an existing facility or 3) create a new facility in an available location. The alternative that produces the most cost reduction is chosen and the algorithm iterates until no further improvements are found.

If the sole criterion is to minimize traveling costs, the resulting partition can be very unbalanced due to the distribution of the sensors in the field. Therefore, the main differences between the problem addressed in this chapter and others found in the literature are 1) The partitioning of the sensor field and positioning of the facilities must guarantee that the service movement of the facilities in their areas is minimized and 2) The workload of the facilities must be optimized and balanced. This work focuses on the capacitated version of the FLP where all the available facilities will be deployed and the number of clients served is bounded and equal for all the facilities. However, there is no restriction on the length of service provided to the clients. Moreover, there are no limitations on the placement of the facilities, no predefined set of possible locations or spare facilities.

In this chapter we propose to extend the clustering algorithm introduced in Chapter 2 to provide a distributed and balanced solution to the FLP to service a network of static sensors by employing mobile maintenance facilities. Our approach combines an initial cluster partitioning, possibly balanced, with an optimization phase to find the optimal placement of the facilities. The final placement will satisfy the condition of load balance among all facilities as well as minimize total facility travel time. The

initial balanced placement, if not present, can be achieved by using any of the algorithms presented in [75] or the "sensor selection" algorithm introduced in Chapter 2. Based on the same design guidelines followed in Chapter 2 and other existing experiences with clustering techniques (e.g., [4, 48]), our goal is again to minimize both the number of messages exchanged by the sensors and the total facility travel time.

As a result of our solution to the FLP, a balanced cluster partition of the network is obtained, where each mobile facility acts as a cluster head. The clusters are optimized through a sequence of iterations or "virtual swapping" of sensors, each time creating a partition whose quality is strictly better than the previous. In particular, in each iteration of the algorithm, sensors are virtually swapped only if doing so improves the quality of the overall solution. Convergence is achieved in a finite number of steps. At this point, cluster membership becomes static and the facility deployment is final.

Our sensor swapping FLP approach is completely distributed and localized. There is no central entity with global knowledge and virtual swapping operations take place only between adjacent areas. Sensors only need to know about their assigned maintenance facility. Experimental analysis shows that the cost in terms of sensor messages remains constant as the network size increases. The quality progression towards convergence and the quality of the final deployment are similar and sometimes better than some benchmark centralized solutions such as K-Means.

3.2 Problem Definition

The instance of the FLP under study makes no distinction on the kind of maintenance task the facilities perform and is based on the following key components:
1) A set of N sensors, $\{S_1, ..., S_N\}$, randomly distributed in an area of unspecified

shape.

2) A set of K mobile facilities, $\{F_1, ..., F_K\}$, initially deployed throughout the area creating a equi-partition of the field. Each cluster or partition, also referred to as a "service area", contains exactly one service facility responsible for the sensors in the cluster.

3) All sensors are static, know their location and are able to communicate with other sensors in their service area and their service facilities. They are also able to monitor their energy levels.

4) The service facilities can move within their service areas. They can communicate with sensors in their areas, also known as "clients", and with facilities in the adjacent areas.

5) The facilities can only serve a predefined number of clients (the same for all facilities), they all have to be deployed and there is no restriction on the eligible locations for deployment.

Under these conditions, the facilities need to improve the initial balanced partition based on a criterion that includes the Euclidean distance to the client sensors as the quality metric. Once the final deployment is achieved, there will be exactly one service facility responsible for each client sensor and the sensors will know the location and identity of their service facility. The sensors may also keep the location of up to two other sensors (i.e., peers) so they can act on their behalf in case of total failure. The number of sensors in each area is balanced and the facilities will be located at a point that minimizes the traveling time within the area.

3.3 The Solution to the FLP

The objective of our solution to the balanced FLP is to find a deployment that facilitates the network maintenance tasks needed to maximize the network lifetime. This

process starts with a balanced partition of the field, which is not necessarily optimal and then applies a series of optimization steps until no further improvement is achieved. Each step or iteration will create a better partition while maintaining the workload balancing requirements. Our solution deals with the refinement of an existing balanced partition and not with the construction of the initial balanced partition itself.

Consequently, once the facilities are relocated at the center of the service area, they will start a series of iterations with the purpose of "trading" the worst client sensor. Each facility will attempt to compress its service area as much as possible. However, in order to give up one of the sensors, another should be accepted from one of the adjacent areas. We call this process "virtual swapping" since there is no actual physical movement of the sensors. Once a trading partner has been found and a new sensor is added to the area, the facilities involved in the trade will compute the center of the new area and move to that location. This process is repeated until no more successful trading partners are found among the adjacent facilities.

3.3.1 Virtual Swapping

Virtual swapping is the process by which sensors from two adjacent areas change their logical membership. There is no actual physical movement involved and the sensors themselves have a quite passive role during this negotiation process. At the beginning of each iteration, each facility will send a SWAP_REQUEST indicating its best candidate sensor for swapping. For example, from all the sensors in the cluster, the facility selects the farthest one. Once the facilities receive a request for swapping they will respond with a SWAP_REPLY indicating a suitable candidate sensor from their cluster based on the same criterion. A facility will respond affirmatively to the

swapping request only if the exchange is mutually beneficial according to the optimization criteria.

At the end of this iteration, the requesting facility has received proposals from all interested adjacent facilities. Then it selects the best candidate among all respondents and replies with a SWAP_ACCEPT message. Once the facilities have agreed on the exchange, they both communicate the change of service facilities to the sensors involved by sending a CHANGE_FACILITY message. This process is repeated until no further improvement can be found (see Figure 3.1 for details).

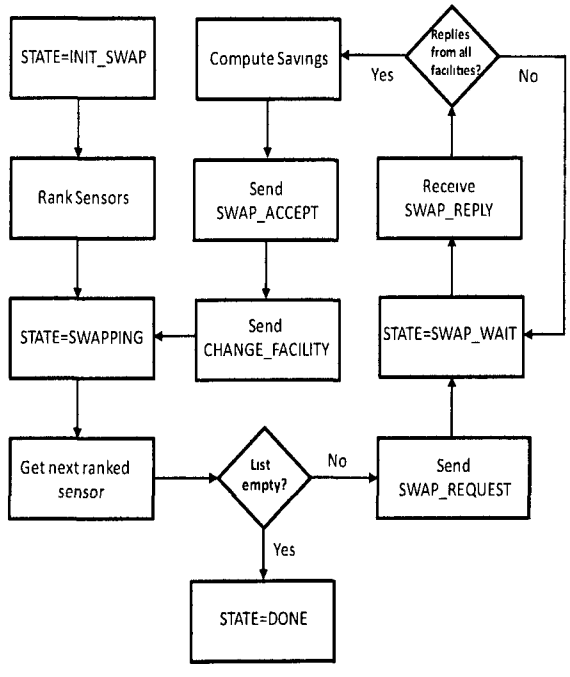


Figure 3.1: Virtual Swapping - Flow Chart.

This optimization mechanism involves the swapping or membership change of some sensors between service areas as long as such operation is mutually beneficial for the adjacent facilities involved. This means that two sensors are considered good

candidates for swapping if, by changing cluster membership, the diameter (i.e., measured in terms of Euclidean distance) of the resulting clusters decreases compared to the original values.

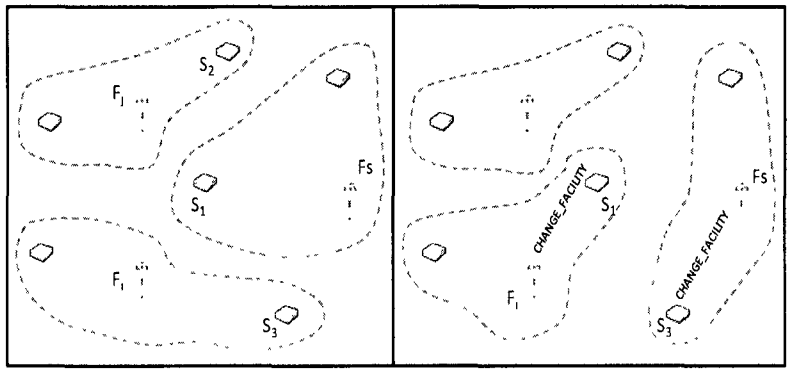


Figure 3.2: Virtual Swapping

Let us examine the following example shown in Figure 3.2. Upon completion of the cluster formation, facility F_s will try to swap or exchange sensor S_1 to minimize its cluster diameter. F_s sends a SWAP_REQUEST(S1) message to all the adjacent facilities and waits for their responses. Only those facilities which will benefit by accepting S_1 will reply with a SWAP_REPLY message containing the proposed sensor. In this example, F_i will respond with SWAP_REPLY(S3) and F_j will send SWAP_REPLY(S2). After receiving responses from all facilities, in this case F_i and F_j, facility F_s will compute the savings for both combinations:

$$Savings(F_i, F_s) = d(F_i, S_3) - d(F_i, S_1) + d(F_s, S_1) - d(F_s, S_3)$$

$$Savings(F_j, F_s) = d(F_j, S_2) - d(F_j, S_1) + d(F_s, S_1) - d(F_s, S_2)$$

The facility that reports the most combined savings will be chosen and a SWAP_ACCEPT($\langle facility \rangle$, $\langle sensor \rangle$) message will be sent to all adjacent facilities. In this example, a simple metric involving only the cluster radius was used. More complex saving functions combining several metrics such as distance, energy levels

and density can also be used However, this may require that facilities collect further information about their clusters and the state of the sensors Regardless of the savings function chosen, the sensor swapping algorithm (see Algorithm 3) describes the basic actions a mobile facility performs based on its status (initially all facilities are in state SWAP_INIT)

The function *FindNextSwapSensor* finds the next swapping candidate based on its ranking value Sensors in the cluster are ranked based on their distance to the facility (i e , farthest to closest) The function *RankSensors*() produces the ranking where the lowest ranked sensor S is such that $d(F,S) = max\{d(F,S_j)\}\ \forall\ S_j$ in the $F's$ cluster *FindNextSwapSensor* returns *null* when all the sensors in the cluster have been explored for potential swapping The function *ComputeSavings* returns the (facility,sensor) pair which provides the most combined savings The sensor swapping process terminates when no further improvements can be found To avoid oscillatory behaviour, the facilities will terminate when they find a previously created cluster or when there are no more successful replies to their swap requests The behaviour of the sensors remains passive throughout this stage, limited only to receive CHANGE_FACILITY messages any time the membership is changed as a result of a successful negotiation

Lemma 3. *The Algorithm Sensor Swapping terminates after a finite number of iterations*

Proof The proof of this lemma follows the same idea as the proof of Lemma 1 presented in Section 2 4 2, since both clustering algorithms have similar termination conditions Let $\{S_1, \ , S_N\}$ a set of N static sensors randomly placed in a plane and $\{F_1, \ , F_K\}$ a set of K mobile facilities also randomly placed in a plane At any

Algorithm 3 Sensor Swapping: Facility F

```
(* In State SWAP_INIT : *)
RankSensors()
rank=1
become SWAP_STATE
(* In State SWAP_STATE : *)
S= FindNextSwapSensor(rank)
if S ≠ null then
    send SWAP_REQUEST(S)
    become SWAP_WAIT
end if
become DONE
(* In State SWAP_WAIT : *)
if receiving SWAP_REPLY(F_i, S_i) then
    numResponses = numResponses+1
    if numResponses = numFacilities then
        (F_b, S_b) = ComputeSavings()
        send SWAP_ACCEPT(F_b, S_b)
        send CHANGE_FACILITY(F_b, S_b)
        currentCluster = UpdateMyCluster()
        if find(currentCluster,ClusterList) then
            send SWAP_DONE message
            become DONE
        else
            UpdateClusterList(currentCluster)
            rank = rank+1
            become SWAP_STATE
        end if
    end if
end if
if receiving SWAP_REQUEST(F_i, S) then
    if ∃ S' ∈ currentCluster where d(F, S') > d(F, S) and d(F_i, S') < d(F_i, S)
then
        send SWAP_REPLY(F,S') to facility F_i
    else
        send SWAP_REPLY(F,NULL) to facility F_i
    end if
end if
```

given iteration of the swapping algorithm, a partition P containing exactly K clusters $\{C_1, ..., C_K\}$ is obtained.

Without loss of generality, we can assume that at the end of the first iteration the algorithm produces the following K clusters $C_1, C_2, ..., C_K$ where $C_i = \left\{F_i, S_{\frac{N}{K}i+1}, S_{\frac{N}{K}i+2}, ..., S_{\frac{N}{K}(i+1)}\right\}$ with $0 \leq i \leq K-1$. Let $P = \{P_1, ..., P_f\}$ be the set of partitions created after each iteration of the algorithm, where $P_i = \{C_1^i, ..., C_K^i\}$ is the cluster partition obtained after iteration i and C_j^i with $1 \leq j \leq K$, are the clusters obtained in such iteration. By definition of the sensor swapping algorithm, P_f is the final partition where no further improvement is achieved. This means that $\forall C_i^f$ $(1 \leq i \leq K)$, $\exists j$ $(1 \leq j < f)$ such that $C_i^f = C_i^j$. The existence of the final partition P_f can be proved by contradiction.

Let us assume that the algorithm does not converge. This means that after each iteration, at least two facilities have been able to successfully trade a pair of sensors. Consequently, a new partition is created where there is at least one cluster C_i^f, $1 \leq i \leq K$, such that $\forall j$, $1 \leq j < f$, $C_i^f \neq C_i^j$. This contradicts the fact that there is only a finite number of combinations of N/K sensors for the same cluster head F_i and hence the number of new partitions is finite. □

3.4 Experimental Analysis

This section examines the simulation results for the sensor swapping algorithm. Once again the simulation software utilized is Omnet++ [77] along with the mobility framework extension [19]. The movement of the service facilities is implemented by the on-demand mobility mode introduced for the simulations of the "sensor selection". Consequently, the facilities will then be able to move on-demand, if needed, after receiving a SWAP_ACCEPT message and the destination will be the computed center of mass based on the senders' original positions. For all the experiments, the facilities and sensors are randomly placed in an area of 1000x1000m^2. Sensors are static

and, once placed, cannot be relocated but the facilities are mobile and there is no restriction on their movements.

Since the main goal of the swapping algorithm is to achieve load balance among the facilities, the experimental analysis centers on two important aspects of the solutions: the quality of the partition and the progression towards the balanced state. The test cases are designed to evaluate the quality of the clustering partition after each iteration and its progression towards convergence.

3.4.1 Quality of the Balanced Partition

The next set of experiments explore the quality of the partition obtained by applying the sensor swapping algorithm. The quality criteria for this case are the average distance of a sensor to its cluster center (i.e., facility) and the average sum of the distances per cluster. The values for the sensor swapping algorithm are compared with two centralized benchmarks: K-Means and Hierarchical Agglomerative Clustering (HAC). Examples of HAC in wireless sensors networks can be found in (e.g., [44, 45, 47, 61, 81]).

Unlike the K-Means algorithm, which starts with a set of pre-defined clusters, the HAC starts with each sensor in one independent cluster. Each iteration of the algorithm merges the existing clusters into new clusters. If no upper bound to the number of clusters is pre-established, the algorithm terminates with all sensors in a single cluster. Another difference with the K-Means and our sensor swapping is that cluster membership is fixed. Once a sensor is assigned to a cluster, its membership will not change. The only possible operation is the merger with an adjacent cluster. For these particular tests, two HAC methods were selected: unweighted pair-group

using arithmetic averages (UPGMA) and Centroid. The UPGMA method calculates the distance between two clusters as the average distance between all pairs of sensors where each member of the pair is taken from each original cluster. In the Centroid method, the distance between clusters is based on the means of the clusters, and the centroid of the merged cluster is the weighted centroid of the two original clusters (see [54] for a more detailed explanation of HAC and K-Means). In our experiments, both HAC algorithms are run until a predefined number of clusters is created (i.e., equal to the number or facilities).

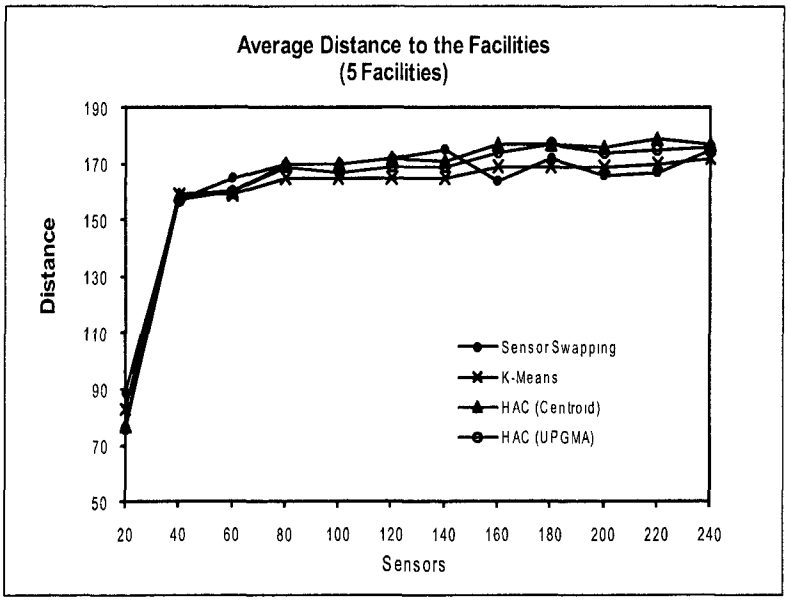


Figure 3.3: Average distance to cluster center comparison for networks of 5 facilities and up to 240 sensors.

Figures 3.3 and 3.4 show the average distance to the cluster center for networks of 5 and 10 facilities with a variable number of sensors. Similar to previous tests, the distributed solution is compared to the centralized benchmarks to evaluate its performance.

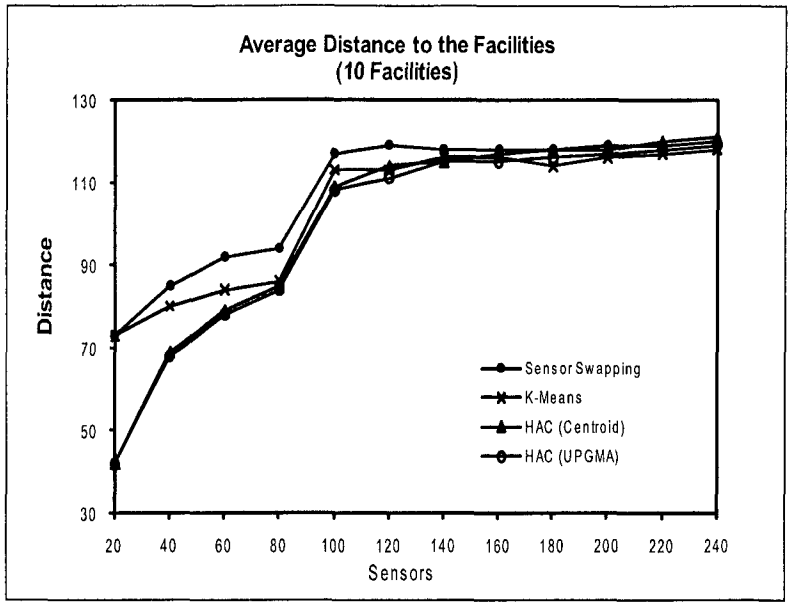


Figure 3.4: Average distance to cluster center comparison for networks of 10 facilities and up to 240 sensors.

Perhaps the most interesting finding resulting from these tests is that, on average, our distributed swapping solution produced partitions of similar quality when compared to the centralized clustering algorithms. For these particular experiments, the same node deployment was used to execute the SPSS K-Means and HAC cluster analysis and the results plotted along with our distributed solution. For both quality parameters, average distance to cluster center and sum of distances (Figure 3.5 and 3.6), the sensor swapping approach and the benchmarks showed comparable results. In particular, as the size of the network increased, the performance of our sensor swapping algorithm got closer to the benchmarks (see Figure 3.4). For larger networks, the benchmarks (e.g K-Means and HAC-UPGMA) also produced clusters with similar average distances but average standard deviation values of 48.82m and 97.72m respectively.

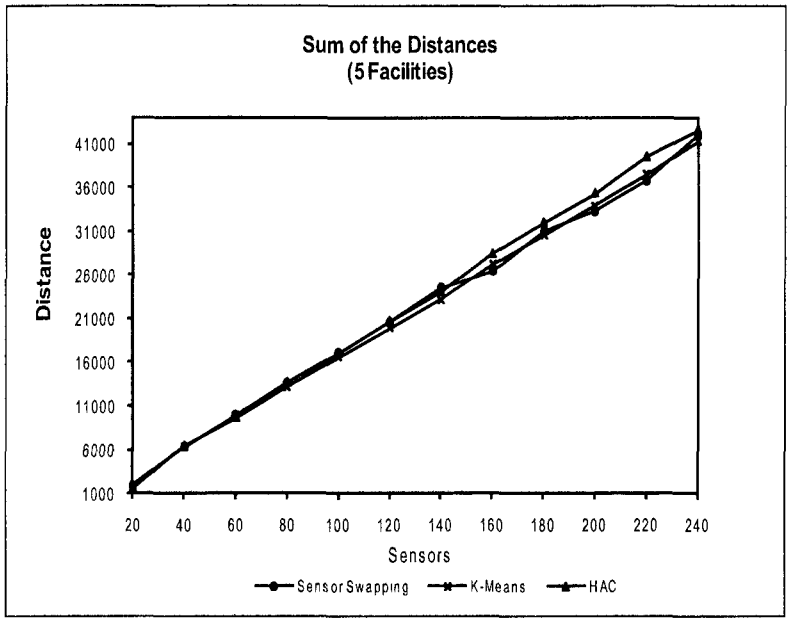


Figure 3.5: Sum of the distances in all clusters for networks with 5 facilities and variable number of sensors.

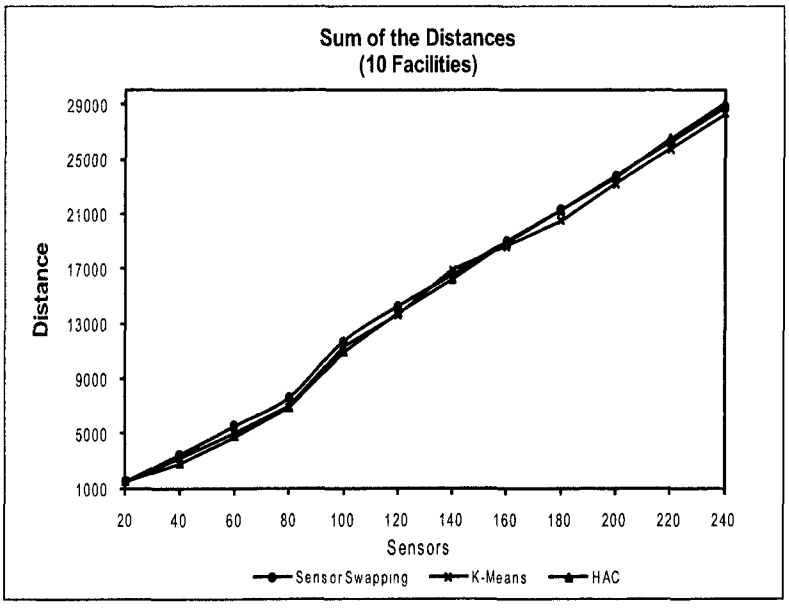


Figure 3.6: Sum of the distances in all clusters for networks with 10 facilities and variable number of sensors.

Despite the similarities between our solution and the benchmarks for the average case, it would be interesting to compare the behaviour of our solution for the worst and best clustering partitions to the benchmarks. Figure 3.7 and 3.8 show the minimum and maximum distances to the cluster center for various random deployments. In particular, for each deployment, the average distance for the best and worst clusters are plotted along with the results for the selected benchmark (i.e., K-Means).

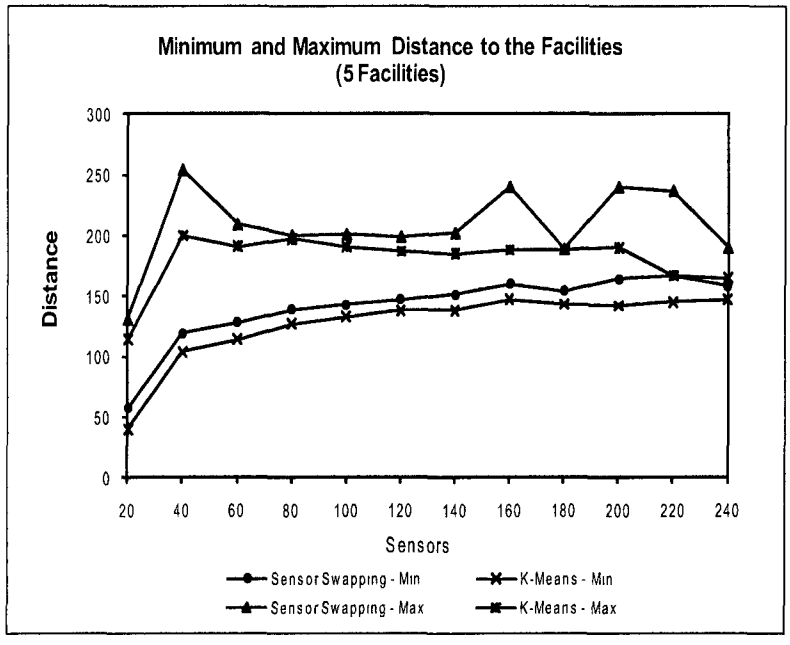


Figure 3.7: Average distance to the cluster centers for networks with 5 facilities and variable number of sensors. Best vs. worst partition.

The quality of the best clusters produced by our sensor swapping algorithm is very close to the benchmark as the size of the network increases. There is more variation for the worst case, but the general trend is to follow the benchmark with better results obtained for deployments with higher sensor-facility ratios.

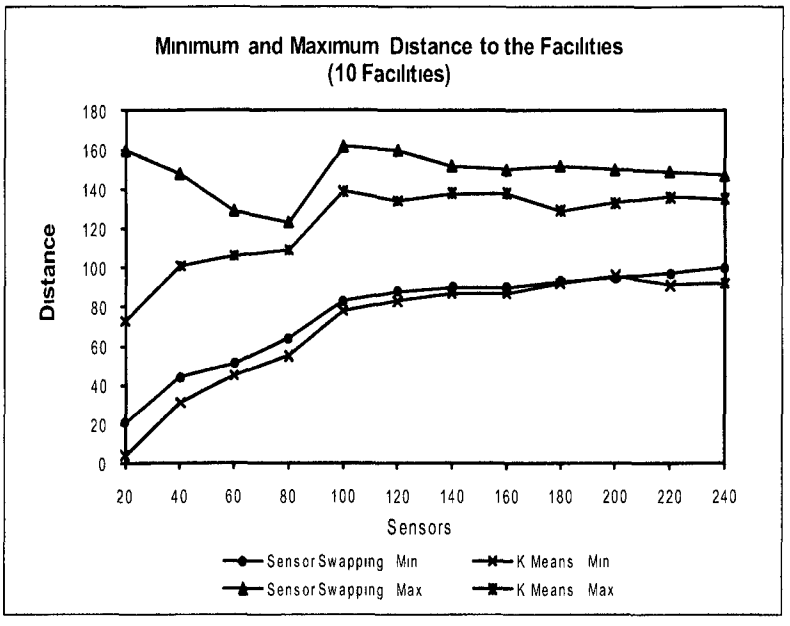


Figure 3 8 Average distance to the cluster centers for networks with 10 facilities and variable number of sensors Best vs worst partition

3.4.2 Progression Towards Convergence

The final set of experiments examine the progression towards convergence and the quality of the partition after each iteration of the algorithm Figure 3 9 and Figure 3 10 show the progression of the average distance to the facilities as well as the facilities' trajectory for each iteration of the algorithm until achieving convergence

The tests show an interesting result, where progression towards the final deployment is not gradual as we may have anticipated On the contrary, after only a few iterations, the variations in the movement of the facilities decrease with an overall trend to only minor adjustments (see Figure 3 10) This behaviour can be seen in both graphic representations For example, in Figure 3 9, each curve represents the changes to the average distance from the cluster center or the current position of the facility (in this case 5 facilities) The behaviour for all 5 facilities is similar, with a rapid decrease of the cluster radius in early iterations and only minor changes thereof

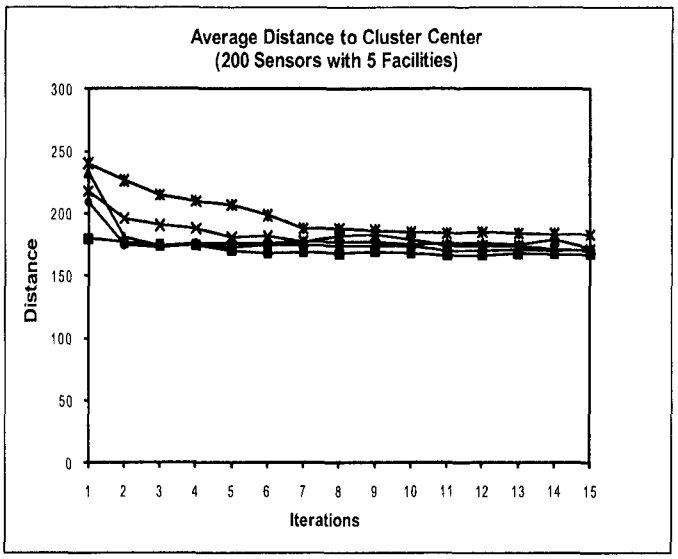


Figure 3.9: Average distance to the facilities after each iteration of the algorithm. Each curve represents a facility and the changes to the average distance in its corresponding cluster.

This is an important feature of our approximation algorithm, which after only a few iterations, will produce a solution close to the one achieved at the convergence point. Depending on the characteristics and requirements of the network, this behaviour could be an advantage in real life applications.

3.5 Conclusions

Throughout this chapter we have discussed the application of an optimization phase to the clustering algorithms proposed in Chapter 2. The ultimate goal of our solution to the capacitated FLP is to extend the life of the network by recharging or replacing depleted or faulty sensors with the help of mobile facilities. The facilities should coordinate their work and achieve this goal by partitioning the network into balanced clusters. This work has also shown that the construction of such a partition in an

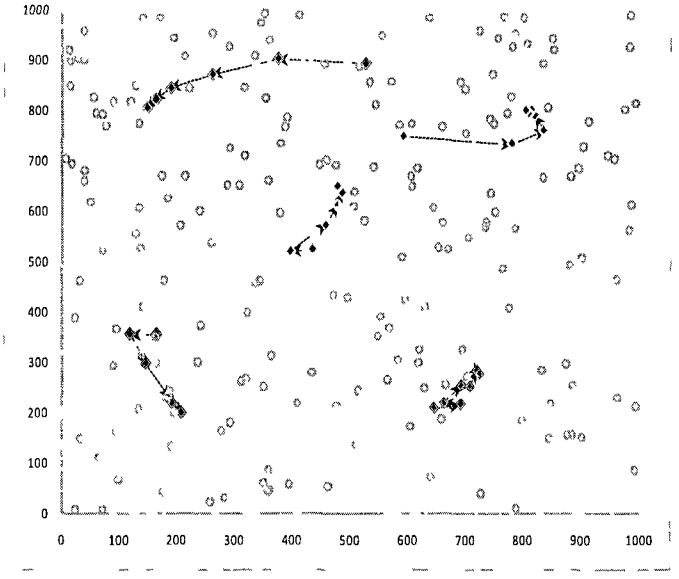


Figure 3.10: Trajectory followed by the facilities after each iteration of the sensor swapping algorithm.

asynchronous environment is possible using the sensor swapping algorithm. This algorithm improves an arbitrary balanced partition by using a series of optimization steps based on a predefined savings function.

The analysis of the several experiments carried out reveals that this simple swapping technique can improve an initial facility deployment with minimal cost in terms of sensor communications. The sensor communication is only limited to receive CLUSTER_CHANGE messages every time a successful negotiation between facilities takes place. In fact, as the size of the network increases, the number of sensor messages remains low with an upper bound of $O(K)$, where K is the number of facilities, since each sensor can change membership at most $(K-1)$ times. Furthermore, progression towards the final deployment is fast (e.g., see Figure 3.10) and the quality of the final partition is comparable to the selected centralized benchmarks.

Chapter 4

Strategies for Mobile Sensor Networks: The Frugal Feeding Problem

4.1 Introduction

In this chapter we examine the problem of energy management in a mobile network environment. The main objective is to maximize the network lifetime, in this case by recharging depleted sensors. For this scenario, a series of service facilities or recharging stations are deployed throughout the sensing area. The main difference compared to the scenarios discussed in Chapter 2 is that service stations are now static. Once deployed, they cannot move, shifting the responsibility to the mobile sensors instead.

In previous studies based on the mobility of certain network components (e.g., [46, 79]), more attention has been given to the base stations as a mechanism to balance the energy levels among all sensors but not for network maintenance tasks. In [46], it is noted that sensors closer to the base station tend to deplete their batteries much faster than other sensors. These sensors have to route/aggregate data flowing from remote parts of the network towards the base station. This disparity creates bottlenecks in areas closer to the base stations. Luo et. al [46] propose the use of mobile base stations to overcome this limitation. They found that, for circular deployments, routes that followed the periphery of the circle, combined with short path routing strategies provided the best overall performance. However, their findings only

increase the lifetime A loss of coverage over time is inevitable since there are no provisions to recharge or replace sensors in the long run

For instances where the mobile sensors are responsible for managing their own energy levels and for coming up with strategies to extend their operating life beyond one battery charge, the standard method to decide when to recharge has been based on fixed thresholds [80] In this case, the service stations take a more passive role and the sensors should be able to compute their remaining operational time and coordinate the use of the service stations [49] Furthermore, for instances where the sensors or robots have to visit a predefined number of points of interests, [80] describes threshold vs non threshold-based solutions where robots decide to visit the service stations depending on their proximity and the nature (locations) of the points of interests

The problem of achieving continuous operation in a robotic environment by refueling or recharging mobile robots has been the focus of attention in recent research papers In particular, this problem is presensted in [41, 42] as the Frugal Feeding Problem (FFP), for its analogy with occurrences in the animal kingdom The FFP attempts to find energy-efficient routes for a mobile service entity, also called "tanker", to rendezvous with every member of a team of mobile robots The FFP has several variants depending on where the "feeding" or refueling of the robots takes place at each robot's location, at a predefined location (e g , at the tanker's location) or anywhere Regardless of which variant is chosen, the problem lies in ensuring that the robots reach the rendezvous location without "dying" by energy starvation during the process

In this chapter, we study the FFP in a wireless sensor network scenario where

mobility capabilities are added to the sensors and static recharge facilities are deployed throughout the sensing area In this variant of the FFP, the responsibility for maintaining the overall health of the network is shifted to the sensor side, whereas the service facilities play a more passive role The rendezvous between sensors and facilities should take place at the closest facility's original position, which is a static location The maximum number of sensors that can rendezvous with a facility at any given time is determined by the number of docking ports or recharge sockets available at the facility

According to the FFP terminology introduced in [41], our problem can be seen as the "tanker absorbed" version of the FFP The rendezvous between the service facility (i e , tanker robot) and the mobile robots (i e , mobile sensors in our case) takes place at the current location of the service facility The location of the service facility is known a priori and the problem is reduced to finding energy efficient routes to reach the facility Another characteristic of our scenario is that the sensors are static in nature That is, they have been deployed and have been assigned specific tasks Therefore, their movement to the rendezvous location will create coverage holes that should be minimized as much as possible

In the FFP, as described in [41], specialized robots (i e , tankers) have to rendezvous with mobile robots to refuel or recharge them The main goal is to minimize the amount of fuel (i e , energy) required to move the robots and tankers to the rendezvous locations The problem can have several variants 1) the rendezvous can take place at the robot's location The robots in need of energy do not move but instead wait for the refueling tanker to come to their rescue This is called the robot-absorbed case 2) the rendezvous takes place at the tanker's location and the robots should move to the tanker's original location This is called the tanker-absorbed case 3)

the rendezvous takes place at locations that do not coincide with the initial robot or tanker locations The FFP also has a combinatorial component pertaining to the order in which the robots should be recharged Finding a solution to the FFP that guarantee that no robots die of energy starvation is an NP-Hard problem (as shown in [41])

The problem of where to place a docking station or recharger is examined by [15] In this case, a team of mobile robots have the specific task of transporting certain items from a pick-up to a drop-off location To be able to work for a prolonged period of time, the robots should interrupt their work and visit the recharge station periodically (i e , tanker-absorbed FFP) Their solution is to place the charger station close enough to the path followed by the robots but without causing interference to the robots' movements

Examples of the robot-absorbed FFP can be found in [5, 18, 63] In all cases, a charger robot is responsible for delivering energy to a swarm of robots The recharging strategy is completely reactive (i e , robots are only recharged when they become out of service and cannot move) In the scenario described in [5], the charger robot is equipped with several docking ports However, the charger robot can travel to recharge a robot in need only if none of the docking ports are occupied, assuming that several depleted robots need to be close by in order to be recharged simultaneously

The solution presented in [18], where a team of miniature robots are deployed along with more powerful docking station robots, is based on the creation of clusters of small robots The decision of which cluster will be serviced first depends on the mean energy level and position of the robots in the cluster Their simulation results

showed that in a network with 64 robots and one charger station with only one docking port, there will be a large number of robots either abandoned or dead due to battery depletion. However, increasing the number of docking ports to 2, affects the performance dramatically by decreasing the number of robot deaths and improving the exploring/dead time ratios.

In [63], static sensors are recharged by chargers or actuators carrying solar panels. Actuators obtain information about the energy distribution in the network by collecting additional information embedded in each communication with the static sensors (e.g., maximum energy, remaining energy, location, etc.). Depending on the topology and the possible presence or not of a single sink, the network is partitioned into clusters. For the single sink case, smaller clusters will be created closer to the sink following the criterion that sensors closest to the sink will deplete their batteries much faster and therefore require more frequent recharges. For cases where there are no sinks, the number of elements in each cluster is balanced. The experimental results with this approach show that a network with 76 sensors deployed in an area of 1000mx1000m requires at least 3 chargers to keep the network alive. The network is considered dead when more than 50% of the sensors die due to battery depletion.

All the aforementioned scenarios satisfy the necessary conditions for mobile robots to be able to recharge themselves as presented in [5]. For example, the robots should be able to monitor their energy levels and detect when it is time to recharge. Second, they should be able to locate and move towards a charging station. Finally, there should be a mechanism for the energy transfer either by docking or plugging into the charging station or via wireless recharging at short distances (e.g., [2, 3, 49]).

In our particular scenario, we consider the sensors to be static in terms of their

sensing requirements In other words, from the point of view of the application (i e , functional requirements), the sensors are static and placed in a specific set of coordinates However, they all have the capability of moving if they decide to go to the service station to recharge their batteries Consequently, the general idea behind our solutions to the network maintenance is to convert the concept of routing into mobility strategies Basically, instead of guaranteeing the delivery of a packet to the intended destination, the sensors now use similar routing techniques to create their own itinerary to reach the service stations

Previous works on sensor localization have shown that the distance between nodes can be estimated by the strength of the incoming signal and the relative coordinates can be computed by exchanging this information between neighbours [10] Also, the sensors could be equipped with a low power GPS receiver to obtain their locations Therefore, in this work we focus on position based routing strategies as the foundation for our proposed mobility strategies

Frey et al [25] provide a detailed survey of position based routing algorithms In particular, there are several identifiable properties of the algorithms that are very useful when evaluating their performance For example 1) Single versus multiple path approaches 2) Avoiding loops the algorithm should not rely on timeouts or keeping information on past traffic as a termination mechanism The algorithms should be loop free, guaranteeing the delivery of the intended packet 3) Distributed operations in a localized routing algorithm each node decides where to send a packet based on its local state, its neighbours and the final destination The objective is to achieve a common goal based on individual efforts without a global knowledge of the network

Position-based routing algorithms can be divided into progress-based and directional algorithms. Examples of progress-based algorithms can be found in [73, 53, 68]. The commonality resides in that they try to forward the packet to a neighbour with positive progress towards the final destination. Positive progress is seen as getting closer to the destination every time the packet is forwarded. There are several variants of progress-based routing and the main difference resides in the selection of the next hop neighbour. In some cases the selection is random, some attempt to send the packet to the neighbour with the most progress within the transmission range, while others select the closest ones. In the directional category, we can find the compass routing proposed in [36], where the next sending node uses the intended destination to calculate its direction and selects, as the next hop, the sensor whose direction is closer to the destination. However, this approach is not loop-free as shown in [68].

Another possible categorization for routing algorithms deals with the number of paths followed. For example, the geographic routing algorithm presented in [31] and the Depth First Search proposed in [69] are examples of single path strategies with guaranteed delivery. Examples of stateless algorithms are presented in [9, 17]. In particular, the Face Routing and GFG (Greedy-Face-Greedy) algorithms construct a planar connected sub graph of the Unit graph. To improve performance, the GFG algorithm switches from greedy to face routing on the Gabriel graph if the node fails to find a neighbour closer to the intended destination.

In our scenario, the sensors need to communicate and coordinate their actions in order to achieve a common goal (i.e., continuous sensing operation without losses due to energy starvation). Furthermore, sensors should coordinate their moves in a loop-free manner so the intended destination (i.e., recharge station) is reached in a finite number of moves or steps. The ultimate goal of the facility absorbed FFP is to reach

a state of energy equilibrium where there are no further sensor losses This chapter will explore the concepts related to position based routing and their application to our particular mobility strategies We will also examine some underlying topologies that guarantee a loop free mobility strategy as well as the network parameters needed to achieve a state of equilibrium

4.2 The Model

The proposed mobility-based energy management approach is built from two main components: mobile capable sensors and static recharge facilities. The general requirement for the model is to extend the network operating life by the autonomous recharge of low energy sensors. However, the ultimate goal is to achieve a state of equilibrium where no further sensor losses are reported and accomplish this with the minimum amount of resources. In general, the model includes the following key components:

1) A set of N sensors, $S = \{s_1, ..., s_N\}$ randomly distributed in an area of unspecified shape.

2) A set of static recharge facilities, $F = \{f_1, ..., f_K\}$ (i.e., rendezvous locations) also randomly distributed throughout the area.

3) Each facility is equipped with a fixed number of recharging ports or sockets. This represents the maximum number of simultaneous sensors at the rendezvous location.

It is assumed that sensors can determine their own positions by using GPS or other localization methods. Sensors can communicate with other sensors within their transmission range R and they all move at the same speed. The distance to the closest facility should be within the sensors' mobility range to guarantee a successful round-trip to the station with one battery charge. All communications are asynchronous; there is no global clock or centralized entity to coordinate communications or actions. The communication environment is contention and error free (i.e., no need to retransmit data) and there is no interference produced by receiving simultaneous radio transmissions (i.e., ideal MAC layer).

The initial placement and relocation of the service facilities can be achieved using

any of the clustering algorithms discussed in Chapter 2. Once the clustering creation is finalized, there will be exactly one recharging station for each sensor in S. The sensors will know the location of their recharging facility, but the facilities are not required to know the number of sensors that will use their resources. With a clustering structure already in place, we can focus on the interactions within a particular cluster. Therefore, without loss of generality, our strategies will be presented in the context of one facility and the subset of mobile sensors assigned to its cluster.

4.3 Passive Approach to Energy Restoration in Mobile Sensor Networks

In this section we attempt to provide the first answer to our initial question: Should the sensors wait or should they act as soon as possible? Let us first start examining the case where the sensors decide to wait. We call this case a passive strategy. In a passive strategy, the sensors will monitor their energy levels using periodic intervals and after any operation (e.g., send, receive, etc.).

The sensors operate in two basic states: BATTERY_OK and BATTERY_LOW. Once the battery level falls below a predefined threshold, which is not necessarily the same for all the sensors and depends on their distance to the station, the sensors will move towards the recharging station. There are two cases to consider: 1) The recharge station is within the sensor's transmission range and the sensor can send a recharge request right away. 2) The recharging station could be outside the sensor's transmission range and a routing mechanism should be in place to forward the recharge request message to the service station.

Alternatively, the sensor could start its journey towards the recharge station and once it gets there (or at least within range) it requests an available socket. Regardless

of the mechanism chosen, the sensor-facility interactions are implemented based on the service station communication pattern shown in Figure 4 1 For simplicity, the pattern shows the case of a service station with only one recharge socket The recharging process is initiated with a RECHARGE_REQUEST sent by a low battery sensor The service station will keep a queue of requests received and a ranking based on the sensors' energy levels When a socket becomes available, the service station sends a RECHARGE_ACCEPT to the smallest ranked sensor (i e , lowest energy sensor) Every time a sensor recharge is completed, the sensor sends a RECHARGE_DONE message to the service station and travels back to its assigned position in the network This process is repeated continuously

The effectiveness of this method depends on several factors such as number of sensors in the cluster, distance to the station, number of recharging sockets, etc Since our ultimate goal is to achieve a point of equilibrium with minimal or not sensor losses, a new question arises will this approach work, and if it does, at what cost? The experimental analysis section provides some of these answers

It is also important to point out that when sensors travel to the stations, they create temporary coverage holes If temporary loss of coverage is an issue of paramount importance for the network, there are solutions to overcome this limitation For instance, the service stations could be equipped with spare sensors The number of spare sensors should be equal to the number of recharging sockets and every time a sensor is accepted (i e , a socket becomes available), a spare is dispatched to the sensor's location to take its place The low battery sensor is now free to travel to the base station and will eventually become a spare after its battery has been recharged

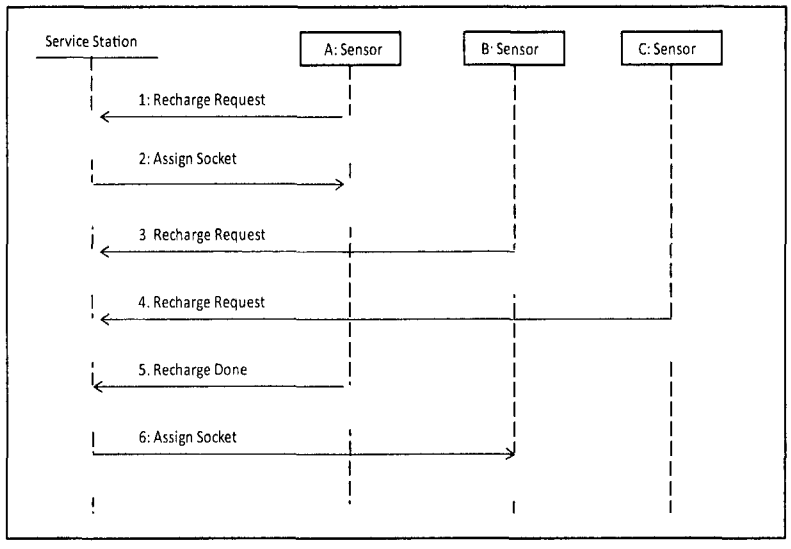


Figure 4.1: Communication pattern for a mutex service station.

4.4 Proactive Approach to Energy Restoration in Mobile Sensor Networks

In this section we examine the case when mobile sensors decide to act before their batteries reach a critical level and a trip to the recharging station is imminent. The general idea is that sensors will try to get closer to their service stations in order capture the so called "front seats". However, the number of front seats is limited (i.e., only sensors within one-hop distance to the station) and since the sensors have responsibilities in their corresponding locations, changing locations cannot be a unilateral decision.

Every time a sensor visits the recharge station, a coverage hole is created. The duration of the hole depends on the recharging time plus the length of the round-trip. In order to minimize coverage holes, sensors will attempt a gradual approach towards the rendezvous location by swapping positions with higher energy sensors. The operating life of a sensor is divided in three stages depending on its battery

status: 1) a BATTERY_OK or normal operation, 2) BATTERY_LOW or energy-aware operation and 3) BATTERY_CRITICAL or recharge-required operation. A sensor in a BATTERY_OK state will perform its regular sensing functions as well as accept any swapping proposal from other sensors with less energy. When the battery level falls below a fixed threshold, the sensor switches its state to a more active BATTERY_LOW state. In this state, the sensor will start its migration towards the service station, proposing swapping operations to sensors with higher energy levels. Finally, a sensor in the BATTERY_CRITICAL state will contact the service station and wait until a socket or docking port has been secured, then it will travel to the station and recharge (see Figure 4.2).

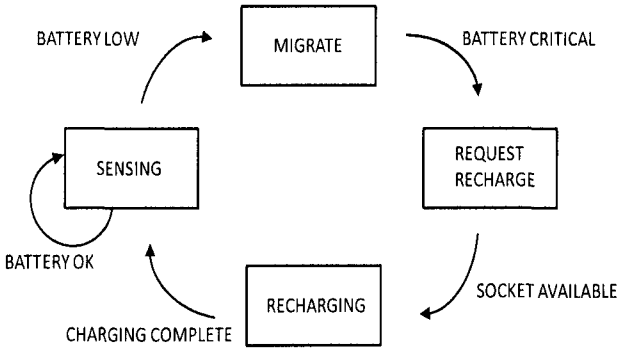


Figure 4.2: A sensor's life cycle.

In this life cycle, it is the migration behaviour that is of interest. The objective of the sensor during migration is to reach the recharge facility in an effective, timely manner, while relying solely on local information. This can be done by allowing the sensor to explore energy-aware routes leading to the recharge facility. The chosen routes are based on a logical Compass Directed unit Graph (CDG).

At this point, we propose to make use of position-based routing strategies. However, instead of sending a packet that needs to be routed until it reaches the intended

target, the sensors have to "route themselves" until they reach the service stations. In particular, we propose to reduce the problem of coordinating the recharging of mobile sensors to the problem of finding energy-aware routes in a CDG built on top of the original topology. The proposed graph incorporates ideas from forward progress routing techniques [6, 25, 60, 68, 84] and the directionality of compass routing [36] in an energy-aware unit sub-graph.

Definition A graph $G = (V, E)$ where the vertices $V = \{v_1, ...v_N\}$ are points in $\mathbb{R}^2$ and edges $E = \{(v_i, v_j)\}$ with $1 \leq i < j \leq N$ is called a Unit Disk Graph (or Unit Graph) if $d(v_i, v_j) \leq R$ where d is the Euclidean distance between the sensors and R is the transmission range.

Definition A graph $G' = (V' \cup F, E')$ with $V' \subseteq V$ and $E' \subseteq E$ is called Compass Directed unit Graph (CDG) if $\forall$ pair of sensors $s_i, s_j \in V'$ and recharge facility F, the edge $s_i \rightarrow s_j \in E'$ iff the following conditions are satisfied:

1. Unit graph criterion: $d(s_i, s_j) \leq R$ where d denotes the Euclidean distance and R is the transmission range.

2. Proximity criterion: $d(s_j, F) < d(s_i, F)$ and $d(s_i, s_j) < d(s_i, F)$

3. Directionality criterion: $\exists s_{jp}$ such that $\vec{s_j s_{jp}} \cdot \vec{s_i F} = 0$ and $d(s_i, s_{jp}) + d(s_{jp}, F) = d(s_i, F)$

Routing algorithms use the hop count as the metric to measure effectiveness. In our case, the hop count would be equivalent to the number of swapping operations between sensors in our CDG. Our solution to the FFP can be divided into two main stages: 1) the construction of the CDG and 2) the incremental swapping approach (i.e., migration) towards the rendezvous location.

4.4.1 Creating the CDG

An example of the proposed CDG for three sensors A,B,C and a facility F is shown in Figure 4.3. In the first stage of the algorithm, it is assumed that all sensors have the required levels of energy to construct the CDG. The process is rather simple and can be summarized by the following actions:

1. Sensors position themselves at some initial fixed location that depends on the task at hand.

2. Sensor A sends a NEIGHBOUR_REQUEST broadcast message inviting other sensors to participate.

3. Upon receiving a NEIGHBOUR_REQUEST message from sensor A, immediate neighbours verify the neighbouring criteria according to the following rules:
   a) Proximity: $d(A,F) > d(B,F)$ and $d(A,B) < d(A,F)$.
   b) Directionality: For example, B and C are neighbours of A if the corresponding projections B_p and C_p on line $\overline{AF}$ intersect the line segment $\overline{AF}$.

4. If both conditions a) and b) are met, then sensors B and C send a NEIGHBOUR_ACCEPT message. Otherwise they send a NEIGHBOUR_DENY message.

In order to save energy, sensor A will then try to deviate as little as possible from the direction of the recharge station F. That is, sensor A will try to minimize the angle $\angle BAB_p$. Therefore, all the sensors that satisfy the conditions a) and b) are ranked according to the following function: $f(s_i, s_j) = \left\{ d(s_i, s_j) + \frac{d(s_j, s_{jp})}{d(s_i, s_j)} \right\}$ where s_i, s_j are the neighbouring sensors, d is the Euclidean distance, F is the recharge station and s_{jp} is the projection of s_j on the line segment $\overline{s_i F}$.

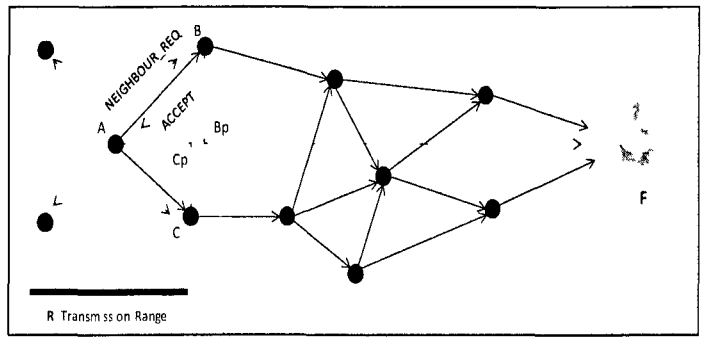


Figure 4 3 Compass Directed Unit Graph

At the end of this phase, each sensor will have two routing tables one containing its children (i e , sensors from which NEIGHBOUR_ACCEPT messages were received) with their corresponding rank and a second table containing its parents (i e , sensors to which NEIGHBOUR_ACCEPT messages were sent) The routing tables are just partial maps of the network indicating the position of the children and parents Algorithm 4 summarizes the behaviour of the sensors during this process The service stations have no involvement at this time The function *DistancePointToLineIn* computes the distance between the potential neighbour and the line segment joining the sensor and the recharge facility If the projection falls inside the segment, the function returns true, otherwise it returns false

The algorithm 4 summarizes the behaviour of the sensors during this process The service stations have no involvement at this time The function *DistancePointToLineIn* computes the distance between the potential neighbour and the line segment joining the sensor and the recharge facility If the projection falls inside the segment, the function returns true, otherwise it returns false

Algorithm 4 GDG Construction: sensor S and facility F

(* In State $INIT$: *)
begin
send $NEIGHBOUR\_REQUEST$ broadcast message
become $BATTERY\_OK$
end
(* In State $BATTERY\_OK$: *)
begin
if receiving $NEIGHBOUR\_REQUEST$ from S' **then**
  if $d(S,F) < d(S',F)$ and $d(S,S') < d(S',F)$ and $DistancePointToLineIn(S,S',F,distanceToLine)$ **then**
    $parentList.Add(S')$
    send $NEIGHBOUR\_ACCEPT$ to S'
  end if
end if
if receiving $NEIGHBOUR\_ACCEPT$ from S' **then**
  $rankingPar = d(S,S')$
  $neighbourList.Add(S', rankingPar)$
  $neighbourList.rank()$
end if
end

4.4.2 Migration Strategy

The second stage of the algorithm starts when sensors change their state from BATTERY_OK to BATTERY_LOW as a result of their battery levels falling below the first threshold Once a sensor enters this state, it will try to get closer to the facility by making a series of one-hop swaps with its graph neighbours

The swapping operation is initiated with a sensor sending a SWAP_REQUEST message to its lowest ranked neighbour Neighbors could be ranked based on their distance (closest to farthest) and their direction relative to the target station Another option of ranking includes the energy levels of neighbours as a metric as well as the energy levels of 2-hop neighbours (i e , children of my children) If the current energy level of the child sensor is larger than the parent sensor, the sensor replies with a SWAP_ACCEPT message and travels to the position of the parent sensor If its energy level is lower, it replies with a SWAP_DENY message Once a requesting sensor has initiated the swapping process it will not entertain any SWAP_REQUEST messages until the swapping operation is completed The swapping operation is considered atomic and once completed both sensors will send a SWAP_COMPLETE message that will be used by current and new neighbours/parents to update their routing tables

The final step of this phase takes place when battery levels fall enough to trigger a change to the BATTERY_CRITICAL state In this state, the sensors behave exactly as in the passive approach and their interaction with the service station is defined by the pattern discussed earlier A BATTERY_CRITICAL sensor sends a RECHARGE_REQUEST message to the recharge station and waits until an available socket is assigned Similar to the passive approach, there are two cases to consider 1) The recharge station is within the sensor's transmission range and 2) The recharging

station is outside the sensor's transmission range and lowest ranked neighbours will forward the request towards the station. If there is no routing mechanism in place, the sensor can initiate its journey (i.e , panic situation) until the station is within range

In an ideal system, all sensors will reach the BATTERY_LOW state when they are exactly at one-hop distance from the rendezvous location When the trip to the recharge station is made from a one-hop position in the graph (i.e., there are no graph neighbours), we call this "*one-hop run*" or "*optimal run*" Contrarily, if the trip is made from any other location, it is called a "*panic run*". We will come back to visit this issue when we discuss the experimental analysis of the different strategies.

Some of the most relevant sensor interactions in the migration stage are summarized in Algorithm 5.

Algorithm 5 Excerpt of the migration algorithm for sensor S to facility F

```
(* In State BATTERY_OK : *)
if receiving BATTERY_CHECK then
    if batteryLevel < BATTERY_LOW_THRESHOLD then
        rank = 1
        become BATTERY_LOW
    end if
    schedule BATTERY_CHECK(interval)
end if
(* In State BATTERY_LOW : *)
if receiving BATTERY_CHECK then
    if batteryLevel < BATTERY_CRITICAL_THRESHOLD then
        send RECHARGE_REQUEST message to facility F
        become BATTERY_CRITICAL
    else
        while rank ≤ numberOfNeighbours do
            send SWAP_REQUEST to sensor with rank: rank
            become WAIT_FOR_SWAP_REPLY
        end while
    end if
    schedule BATTERY_CHECK(interval)
end if
(* In State WAIT_FOR_SWAP_REPLY_STATE : *)
if receiving SWAP_ACCEPT from S_i then
    move to S_i
    neighbourList.update()
    nieghboutList.rank()
    send SWAP_COMPLETE
    rank=1
    schedule BATTERY_CHECK(interval)
    become BATTERY_LOW
end if
```

Algorithm 6 Migration Algorithm - Part 2

```
if receiving SWAP_DENY from S_i then
    rank = rank + 1
    schedule BATTERY_CHECK(interval)
    become BATTERY_LOW
end if
(* In State BATTERY_CRITICAL *)
if receiving RECHARGE_ACCEPT then
    lastPosition = currentPosition
    move to Facility
    schedule BATTERY_CHECK(interval)
    become RECHARGING
end if
if receiving SWAP_REQUEST(S', senderBatteryLevel) then
    if batteryLevel > senderBatteryLevel then
        send SWAP_ACCEPT
        move to S' location
        neighbourList update()
        neighboutList rank()
        send SWAP_COMPLETE
    end if
    schedule BATTERY_CHECK(interval)
    become BATTERY_LOW
end if
```

Algorithm 7 Migration Algorithm - Part 3

```
(* In State RECHARGING *)
if receiving BATTERY_CHECK timer then
    if batteryLevel = MAX_BATTERY_LEVEL then
        send RECHARGE_DONE(Facility)
        travel to last_position
        send SENSOR_RECHARGED message
        become BATTERY_OK
    else
        schedule BATTERY_CHECK(interval)
    end if
end if
(* In any State BATTERY_OK, SWAPPING,
WAITING_FOR_SOCKET *)
if receiving SWAP_COMPLETE(S') then
    if neighbourList Find(S' coodinates) then
        neighbourList update()
        neighboutList rank()
    end if
    if parentList Find(S' coodinates) then
        parentList update()
    end if
end if
if receiving SENSOR_RECHARGED(S') then
    if neighbourList Find(S' coodinates) then
        send PARENT_UPDATE message to S'
    end if
    if parentList Find(S' coodinates) then
        send NEIGHBOUR_UPDATE message to S'
    end if
end if
```

4.4.3 Properties of the CDG

There are two important properties of the CDG (i e , dynamic and self-correcting) that can be explained by the following scenarios Both scenarios may cause situations where the information in the neighbouring tables is obsolete

- Scenario 1 Simultaneous swapping As part of the swapping process, the participating sensors exchange their neighbouring information, that is, their corresponding children and parent tables However, since multiple swapping operations may occur at the same time, when a sensor finally arrives at the position occupied by its swapping partner, the information in its neighbouring tables may be out-of-date

- Scenario 2 Sensor recharging While this process takes place, other sensors may be swapping positions Once the recharging process is finished, the sensor returns to its last known position However, the structure of the network around it has changed This situation is even more evident when trips to the facility are made from distances of more than one hop as a result of "panic runs"

The left side of Figure 4 4 shows two concurrent swapping operations between sensor $S2 \leftrightarrow S3$ and $S4 \leftrightarrow S5$ respectively As part of the swapping process, the sensors involved exchange their routing information, that is, their corresponding neighbour and parent tables However, since multiple swapping operations may occur at the same time, when sensor S2 finally arrives to the position occupied by S3, it believes (i e , according to its routing table) that S4 is one of its neighbours However, this is no longer the case since S4 has switched positions with S5

The other situation takes place when a sensor finally makes a trip to the recharge station The right side of Figure 4 4 depicts the case when two sensors, S1 and S7, are being recharged simultaneously While this process takes place, sensors S2 and

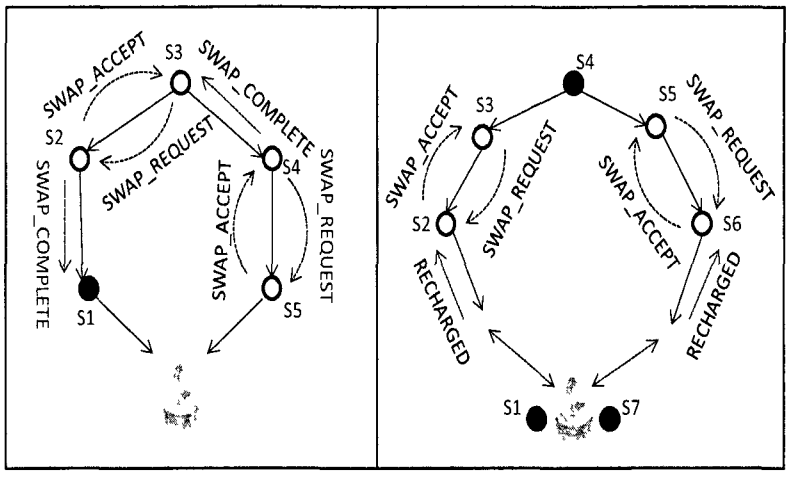


Figure 4.4: Two common swapping scenarios

S3, and sensors S5 and S6 are swapping positions. Once the recharging process is finished, sensors S1 and S7 return to their last known position. However, the structure of the network around them has changed. This situation is even more evident when trips to the service station are made from distances of more than one hop as a result of a panic run.

The solution to these problems is to define the neighbouring information as position-based tables, where the important factor is the relative position of the neighbours and not their corresponding IDs. The information of the actual sensors occupying the positions is dynamic. In other words, a sensor in a given position (x, y) knows that at any given point in time it has n children at positions $(x_1, y_1)...(x_n, y_n)$ and p parents at positions $(x'_1, y'_1)...(x'_p, y'_p)$. This information is static with respect to (x, y) and will not be modified. However, the identity of the sensors occupying the positions is dynamic and will get updated every time a swapping operation occurs. The mechanism to detect changes in the routing tables is triggered by sending a SWAP_COMPLETE

message. When two neighbouring sensors successfully complete a swapping operation, they will announce their new positions by sending SWAP_COMPLETE messages. Sensors within the transmission range that listen to this message will verify whether any of the positions involved in the exchange belong to their routing tables and update the appropriate entry with the ID of the new occupant of that position.

On the other hand, a sensor returning from the service station (e.g., scenario 2) needs to re-discover the new occupants of its routing tables. This process is initiated by a SENSOR_RECHARGED message sent by the newly recharged sensor as soon it reaches its last known position on the network. Potential children and parents, upon receiving this message, will reply with CHILD_UPDATE and PARENT_UPDATE messages accordingly. This process is also used for parents to update their information about the energy levels of this newly recharged sensor.

These two important properties, along with a neighbouring criteria that incorporates ideas from forward progress and compass routing [25, 36, 68] in an energy-aware unit graph, ensure the following lemma:

Lemma 4. *The swapping-based proactive solution to the FFP guarantees that all sensors reach the rendezvous location within a finite number of swapping operations.*

Proof. Let $G = (V, E)$ be a CDG with a set of vertices $V = \{s_1, ..., s_N, F\}$ where s_i, $1 \leq i \leq N$ represent sensors and F denotes the rendezvous location. Let E be a set of edges of the form $s_i \rightarrow s_j$ where s_j is a CDG neighbour of s_i.
Without loss of generality, we can assume that for any path $P_i =< s_i, ..., s_k, F >$ leading to the recharge station F, with $1 \leq i < k \leq N$, the sub-path containing the sensors $< s_i, ..., s_k >$ does not contain any cycles. By contradiction, let us assume that the rendezvous location cannot be reached. This means that at some point

during the execution of the algorithm a given sensor finds itself in a loop (i.e., a cycle C of arbitrary length L is found). Let $C = \{s_i, s_{(i+1)}..., s_{(L-1)}\} \bigcup \{s_L, s_i\}$ with $1 \leq i < L \leq N$. If such a cycle C exists, sensor s_i must be graph neighbour of sensor s_L which means that $d(s_i, F) < d(s_L, F)$. This contradicts the proximity criterion (2)(triangular inequality). Hence, the Lemma holds. □

4.4.4 Extreme Cases

So far, the proactive strategy seems not only possible but intuitively more efficient than a passive approach. However, for some specific deployments, the proactive solution may not report any improvements over the passive approach. The deployment shown in Figure 4.5 shows the trajectory followed by a sensor s_1 during its migration towards the facility F.

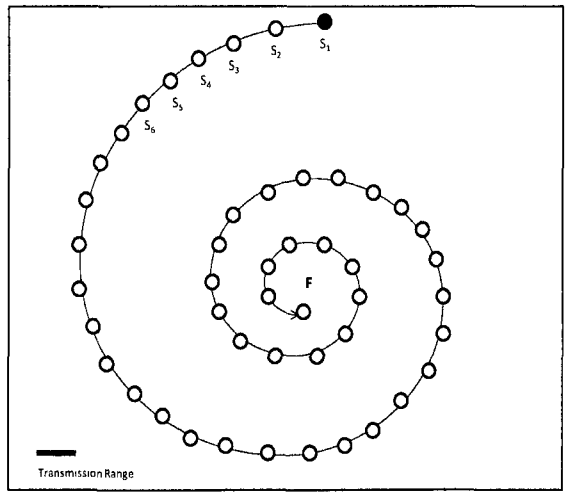


Figure 4.5: Proactive strategy for Sensor s_1 in a spiral deployment.

If a proactive strategy is selected for this particular deployment, the sensor will start a gradual approach towards the facility. The sensors' limited transmission range implies that only one neighbour will be discovered during the CDG creation. That is, $\forall$ sensor $s_i \in \{s_1, s_2, ...\}$, s_{i+1} is graph neighbour (i.e., child node) of s_i (i.e.,

$d(s_{i+1}, F) < d(s_i, F)$) and s_{i-1} its corresponding parent. The locally-based swapping selection criteria will force sensor s_1 to exchange positions with its only available graph neighbour s_2. Consequently, s_1 will take the longest possible path to rendezvous with the facility.

In this particular example the proactive strategy incurs an excessive and unnecessary waste of energy by a continuous sensor swapping. The sensor will eventually reach a BATTERY_CRITICAL state and will default to a passive behaviour. However, this could have been avoided by taking a passive approach and waiting in its original position until the BATTERY_CRITICAL state is reached. In this particular deployment following a passive approach would have maximized sensing time by avoiding temporary coverage holes due to unnecessary swapping operations.

4.5 Experimental Analysis

This section examines the simulation results for the passive and proactive strategies described in the previous sub-sections. The simulation software utilized was Omnet++ [77] along with the mobility framework extension [19]. For all the experiments, the sensors and facilities are randomly placed in an area of 1000x1000m^2 and the sensor transmission range is set to 100m. The analysis centers on three important aspects of the solutions:

1) Whether or not a state of equilibrium is achieved, and the number of sensor losses until such condition is met.

2) The quality of the strategy, measured in terms of optimal runs vs. panic runs.

3) The resources required to achieve a perfect state of equilibrium.

For all the simulation scenarios, constant cost values were assigned to each basic

operation: send, receive, idle and locomotion. The relationship between these values follows some of the experiences found in the literature [21, 22]. The sensors will check their battery status at periodic intervals and after an event has occurred (e.g., a new message is received, etc.). The intervals are chosen randomly and simulate the sleep-idle-active cycle normally followed by the sensors. Every time the battery is checked, the levels are decreased by a predefined constant. This particular behaviour simulates the energy consumption in the idle state. The energy used when receiving information will be 50% of the energy (E) required to send a message. The locomotion cost is based on the weighted Euclidean distance traveled with a weight factor of $\frac{1}{5}E$ for each unit of distance (e.g., meters). The focus of the experiments is not to measure the energy consumption in each operational state but to establish similar parameters to evaluate and compare the performance among the proposed strategies.

4.5.1 Sensor Losses Over Time

The first set of experiments attempt to find out whether the active solution reaches a state of equilibrium. In other words, the objective is to measure the number of sensor losses due to battery depletion over time until the system reaches a state where no further sensor losses are reported. In this context, several proactive strategies are examined: 1) The closest-first strategy, where sensors attempt to make forward progress by swapping positions with the closest neighbour (i.e., default ranking). 2) Variable degree strategy, where the number of neighbours is restricted and an upper bound on the graph degree is set from single path (degree 1) until degree 4. The neighbour selection is similar to the closest-first but establishes an upper bound, and finally 3) the closest-with-most-energy-first strategy, where the sensor selects the swapping partner based on the distance/energy ratio of its neighbours.

The results of an experiment involving 100 sensors and one service facility are shown in Figure 4 6 The facility is equipped with two sockets, allowing two sensors to be recharged simultaneously A series of 30 tests with different random deployments are run for 10^6 simulation seconds The sensor transmission range is fixed at 100m and the energy ratio for sending/receiving a packet is set to a constant (E $E/2$) Locomotion costs were based on the weighted Euclidean distance with a weight factor of $\frac{1}{5}E$ per meter traveled The results show that all the variations of the pro-active approach reached the state of equilibrium This means that all the energy spent during the graph creation, swapping and graph reconfiguration in a network with a 100 1 sensor-facility ratio with two sockets did not overwhelm the system to the point of preventing it from reaching equilibrium

In comparison, for a similar network size, the solutions presented in [18] and [63] required 2 and 3 stations, or actors, respectively, to maintain a live network (i e , 50% or more sensors remain after equilibrium was reached) In our case, equilibrium was achieved with 1 facility with two docking ports for a similar network size and over 80% of network survivability

Another interesting result is that graph degree had a positive impact in the performance of the algorithms Multiple path approaches outperformed single-path strategies even when the number of control messages was higher The closest-first appears to be the best performer of the group Another interesting observation is that the closest-with-most-energy-first approach did not provide the best results Contrary to what we may have anticipated, the idea of adding the energy level in the ranking did not report great improvements A possible explanation for this behaviour is that neighbours with higher energy levels were favored over others closer in directionality but with relatively lower energy levels

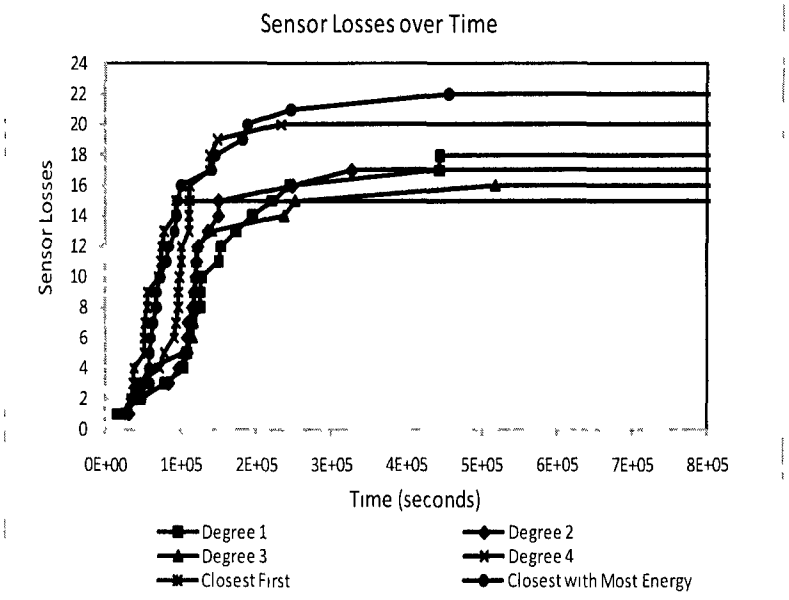


Figure 4.6: Failures over time for pro-active strategies.

The next logical question could be: how do proactive strategies perform when compared to a passive approach? The second part of this test (Figure 4.7) addressed this issue by comparing two proactive strategies: closest-first and the single path strategy, with the passive approach. Surprisingly, even the single path proactive strategy outperformed the passive approach by a significant margin. Even though the passive strategy reached the state of equilibrium faster than the single-path pro-active strategy, the cost in terms of sensor losses was very high. This result implies that if a passive approach is chosen for high sensor-facility ratio deployments, the number of recharge sockets in this experiment is too restrictive. This result is similar to the passive approach followed in [18] where they noticed a significant improvement by adding a second recharge station.

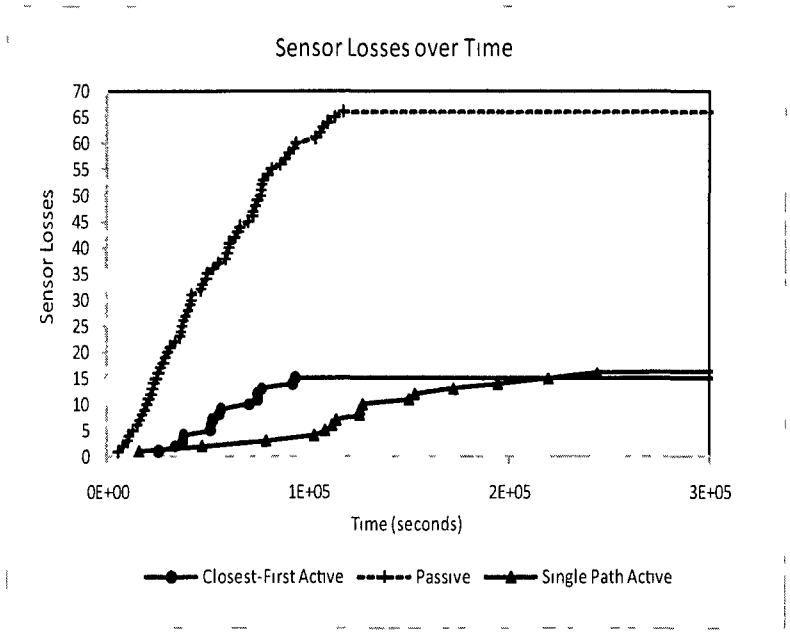


Figure 4.7: Passive Strategy vs. Proactive Strategies

4.5.2 Quality of the Solution

The second set of tests is designed to verify the quality of the proactive solutions. In an ideal system, sensors following a proactive strategy should reach the state of equilibrium using one-hop runs only. This test examines the breakdown between one-hop and panic runs for all the proactive solutions. The experimental setup is the same as the previous case. The breakdown between one-hop and panic runs is shown in Figure 4 8 . As expected, the graph degree has a positive impact on the quality of the solution. As the node degree increases, there are more alternative paths to get to the service station and ultimately more "front seats" available. The closest-first approach, which had no limitation on the number of neighbours within range, is once again the best performer among all the strategies, with 40% of all the trips being optimal runs.

This set of experiments exposes an interesting property of the network and the

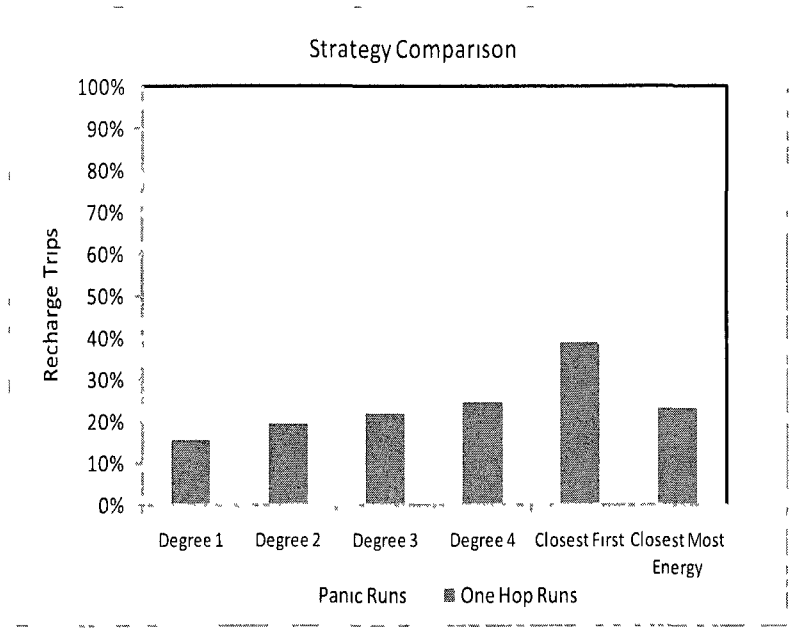


Figure 4.8: One-hop runs vs. Panic runs.

location of the recharge station. In all the experiments, the service station was located at the center of the area. The density of the graph around the service station, in conjunction with a multiple path and unrestricted degree strategy, such as closest-first should yield the best results. To maximize the number of sensors within one-hop distance to the recharging station, the stations could be deployed in the denser areas of the network. On the other hand, from a practical point of view, an approach that reaches a perfect state of equilibrium faster and with fewer resources should be preferred regardless of the breakdown between optimal and panic runs.

4.5.3 Achieving Perfect Equilibrium

The last set of tests attempts to determine the facility resources needed to achieve a perfect state of equilibrium; that is a state of equilibrium with no sensor losses due to battery depletion. To illustrate the experiment, the best proactive approach (i.e.,

closest first) is selected and compared to the passive solution The experiment involved a series of simulations in a network with 100 sensors and 1 service station but varying the number of recharge sockets. Figure 4.9 shows the comparison between the two solutions and plots the impact of the number of recharging sockets on the total number of sensor losses until equilibrium.

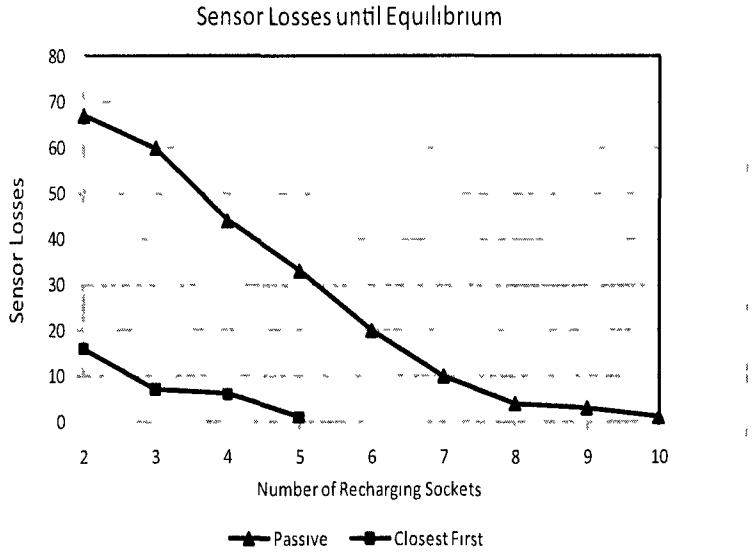


Figure 4.9: Failures until equilibrium by number of recharging sockets

A positive and expected result of this experiment was that both approaches reached the state of perfect equilibrium at some point. The main difference though, is that the passive approach needed twice as many sockets to eliminate all losses. On a positive note, the progression for the passive approach is rather fast considering the high number of failures with only two sockets.

4.6 Conclusions and Summary of Experimental Results

Throughout this chapter we have examined the problem of energy management in a mobile network scenario, focusing on networks with mobile sensors and static service facilities Existing energy management approaches are mainly reactive and based on fixed thresholds as the main deciding factor in the strategy to follow However, the ultimate question in this situation is what should the sensors do? Should they wait or should they take a more proactive approach? The results of this chapter provide some answers to these issues by comparing passive vs proactive approaches to energy management based on different mobility strategies

Our solution recommends taking a more proactive approach to energy restoration based on several mobility strategies In particular, we proposed to reduce the problem of coordinating the recharging of mobile sensors to the problem of finding optimal routes in a logical Compass Directed unit Graph (CDG) built on top of the original topology

In summary, the proposed proactive solutions have the following properties 1) The proposed CDG guarantees that sensors will reach the recharge facilities in a finite number of swapping operations The trajectory is loop-free 2) All decisions made by the sensors regarding the next swapping operation are based on local knowledge (i e , the algorithms are completely distributed and localized) 3) New sensors can be added or deleted at any time and new neighbours are re-discovered any time a successful swapping or recharge operation takes place, making the graph dynamic and self-correcting

Any successful energy management strategy must reach a state of equilibrium,

where no further sensor losses are reported and sensors cooperate to share a common recourse (i.e., recharge station). To measure the quality of the solutions we centered our analysis on several key indicators, such as number of sensor losses until equilibrium, distance traveled to reach the service station (i.e., optimal runs vs. panic runs) and resources needed to achieve a perfect equilibrium (i.e., no sensor losses due to battery depletion). The experimental results showed that:

1) All the variations of the pro-active approach (i.e., closest-first, variable degree, closest-with-most energy) reached the state of equilibrium.

2) The closest-first proactive strategy outperformed all other proactive strategies.

3) Even the single path proactive strategy outperformed the passive approach.

4) The closest-first strategy provided the most balanced solution, where 40% of the recharge trips were initiated from a one-hop distance to the service station.

5) All proactive solutions reached the state of perfect equilibrium by increasing the number of recharging sockets. However, the passive solutions needed twice as many sockets when compared to the closest-first active strategy.

Chapter 5

Further Analysis of Proactive Strategies for the Frugal Feeding Problem

5.1 Introduction

In this chapter we examine in more detail the proactive solution to the facility-absorbed FFP introduced in Chapter 4. In particular, we intend to take the simulation results presented in Chapter 4 a step further by studying the performance of our proactive algorithms under the following scenarios:

1. Variable transmission range values. These scenarios will help in the simulation of different sensor technologies and protocols (e.g., 802.11, 802.15.4, etc.) and their direct impact on the neighbor selection process and number of sensor losses until equilibrium.

2. Variable mobility cost. For example, assigning different relative cost values to each unit of distance traveled would help determine the limits of the proposed proactive strategies.

3. Explore new underlying topologies based on a different neighbor selection process. Is it possible to be more selective when choosing the graph neighbors? Is having fewer but better selected neighbors a more successful strategy?

4. Enhance sensor knowledge by adding information about the energy levels of the 2-hop graph neighbors. Is more knowledge better to achieve energy equilibrium? This added knowledge may impact the path selection process and facilitate the

migration through higher energy areas of the network Is the added cost derived from keeping more sensor information worth it?

5 Explore the role of the recharge facility Is it really necessary to fully recharge a sensor before injecting it back into the network?

The next sections will elaborate on each specific scenario and provide some answers to these questions

5.2 Exploring Different Topologies

In this section we examine the impact of the underlying topology on the overall performance of the proactive strategies The idea is to evaluate the same mobility strategies studied in Chapter 4 under a different underlying topology The requirements for the new topology remain the same, and they are

1) It should be built using local information only

2) It should be flexible enough to operate in an asynchronous environment

3) It should be dynamic, self-correcting

4) Mobility strategies based on this topology should be loop-free

From the experimental results presented in Chapter 4, we concluded that the number of graph neighbours has a direct impact on the performance of the proactive solution Having more immediate graph neighbours implies more options when exploring a greedy migration towards the recharge station but it also means more interactions, notifications, etc , as more sensors will be affected by SWAP_COMPLETE and SENSOR_RECHARGE messages The problem is to determine the right number of sensors within range that should be selected as neighbours Here, we have a clear trade-off between flexibility when choosing a migration path and the required maintenance overhead

For the new topologies to consider, the sensors will select their graph neighbours based on the concept of Gabriel neighbours and Relative neighbours Two points A and B are said to be Gabriel neighbours if their diametric circle does not contain any other points A graph where all pairs of Gabriel neighbours are connected with an edge is called the Gabriel graph In our case, two sensors s_1 and s_2 with coordinates (x_1, y_1) and (x_2, y_2) are Gabriel neighbours if the circle with center $(\frac{x_1+y_1}{2}, \frac{y_1+y_2}{2})$ and

radius $\frac{d(s_1,s_2)}{2}$ does not contain any other sensor. A particular case of a Gabriel Graph is the Relative Neighbor Graph where sensors s_1 and s_2 are relative neighbours if there are no other sensors in the Lune between sensors s_1 and s_2. That is, if $\forall\, S$, $S \neq s_1$ and $S \neq s_2$, $d(s_1, s2) < max\,\{d(s_1, S), d(s_2, S)\}$ where d denotes the Euclidean distance between two sensors. [71, 32]. Figure 5.1 shows examples for each particular case.

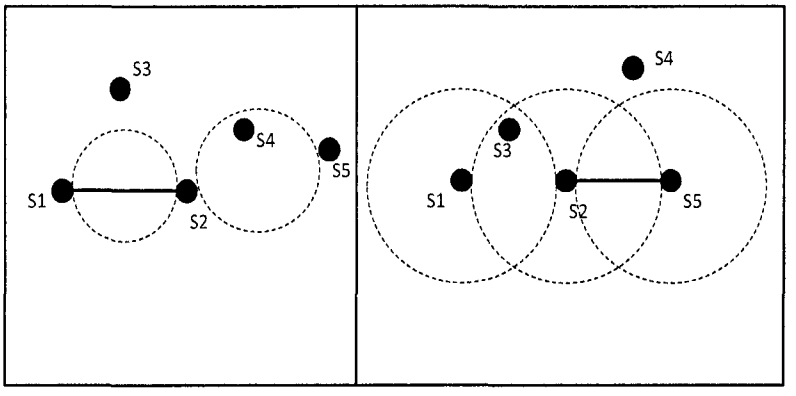


Figure 5.1: Gabriel neighbours (left) and Relative neighbours (right)

In this new scenario where low energy sensors will select their Gabriel or Relative neighbours as the potential swapping partners, the migration strategy towards the recharge station will be based on finding energy efficient routes on a Compass Directed Gabriel Graph (CDGG) or a Compass Directed Relative Neighbor Graph (CDRNG).

Definition Let $G = (V, E)$ be a Unit Disk Graph with vertices V and a set of edges E. A graph $G = (V' \cup F, E)$ with $V' \subseteq V$ and $E' \subseteq E$ is called *Compass Directed Gabriel Graph* (CDGG) if $\forall$ pair of sensors $s_i, s_j \in V'$ and recharge facility F, the edge $s_i \rightarrow s_j \in E'$ iff the following conditions are satisfied:

1. Unit graph criterion: $d(s_i, s_j) \leq R$ where d denotes the Euclidean distance and R is the transmission range.

2. Proximity criterion: $d(s_j, F) < d(s_i, F)$ and $d(s_i, S_j) < d(s_i, F)$

3. Directionality criterion: $\exists s_{jp}$ such that $\vec{s_j s_{jp}} \cdot \vec{s_i F} = 0$ and $d(s_i, s_{jp}) + d(s_{jp} F) = d(s_i, F)$

4. Gabriel neighbour criterion: $\nexists s_k \in V'$ such that $d(s_k, \frac{s_i + s_j}{2}) < d(s_i, \frac{s_i + s_j}{2})$

Definition Let $G = (V, E)$ be a Unit Disk Graph with vertices V and a set of edges E. A graph $G = (V' \cup F, E)$ with $V' \subseteq V$ and $E' \subseteq E$ is called Compass Directed Relative Neighbor Graph (CDRNG) if $\forall$ pair of sensors $s_i, s_j \in V'$ and recharge facility F, the edge $s_i \rightarrow s_j \in E'$ iff the following conditions are satisfied:

1. Unit graph criterion. $d(s_i, s_j) \leq R$ where d denotes the Euclidean distance and R is the transmission range.

2. Proximity criterion: $d(s_j, F) < d(s_i, F)$ and $d(s_i, s_j) < d(s_i, F)$

3. Directionality criterion: $\exists s_{jp}$ such that $\vec{s_j s_{jp}} \cdot \vec{s_i F} = 0$ and $d(s_i, s_{jp}) + d(s_{jp} F) = d(s_i, F)$

4. Relative neighbour criterion: $\nexists s_k \in V'$ such that $d(s_i, s_k) < d(s_i, s_j)$ and $d(s_k, s_j) < d(s_i, s_j)$

5.2.1 Creating the CDGG and CDRNG

Figure 5.2 shows an example of the proposed CDGG for three sensors A,B,C and a facility F. In the first stage of the algorithm, it is assumed that all sensors have the required levels of energy to construct the CDGG. The process can be summarized by the following actions:

1. Sensors position themselves at some initial fixed location that depends on the task at hand.

2. Sensor A sends a NEIGHBOUR_REQUEST broadcast message inviting other sensors to participate.

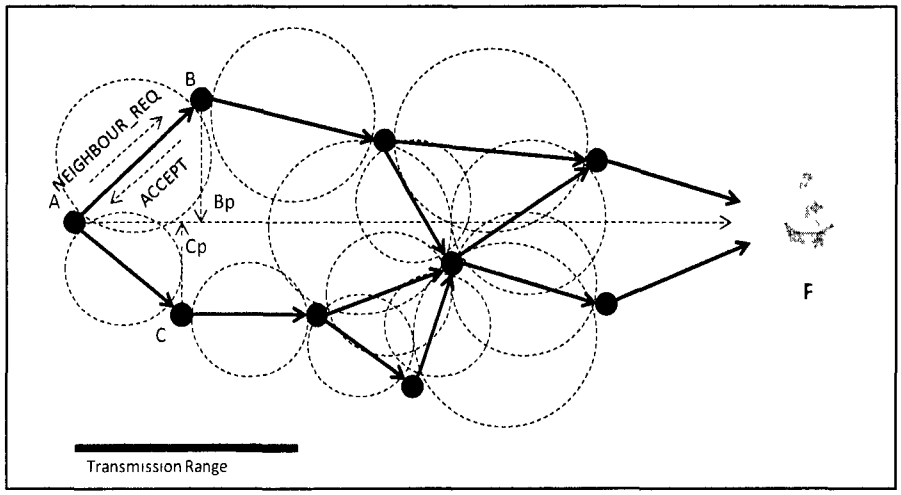


Figure 5.2: Compass Directed Gabriel Graph

3. Upon receiving a NEIGHBOUR_REQUEST message from sensor A, immediate neighbours verify the neighbouring criteria according to the following rules:
   a) Proximity: $d(A,F) > d(B,F)$ and $d(A,B) < d(A,F)$.
   b) Directionality: For example, B and C are neighbours of A if the corresponding projections B_p and C_p on line $\overline{AF}$ intersect the line segment $\overline{AF}$.

4. If the conditions a) and b) are met, then sensors B and C send a NEIGHBOUR_ACCEPT message. Otherwise they send a NEIGHBOUR_DENY message.

Up to this point, the process is the same as the creation of the CDG introduced in Chapter 4. However, to guarantee that only the Gabriel neighbours are selected as graph neighbours, the sensor should implement the following actions:

1. Upon receiving a NEIGHBOUR_ACCEPT message from a potential Gabriel neighbour S', the receiving sensor S verifies if there is already a graph neighbour in the disc with center $(\frac{S_x+S'_x}{2}, \frac{S_y+S'_y}{2})$ and radius $\frac{d(S,S')}{2}$. If such a neighbour exists, then sensor S sends a NEIGHBOUR_DENY message to S'.

2 If no existing graph neighbour is found in the previous step, this means that sensor S' is in fact a Gabriel neighbour However, some of the existing graph neighbours could be affected by this newly accepted sensor and they are no longer Gabriel neighbours If the newly accepted sensor S' falls in the diametric disc between sensor S and one of the existing graph neighbours S_i, the neighbour in question should be excluded by sending it a NEIGHBOUR_DENY message

The creation of the CDRNG follows the same pattern with only one minor change to verify the relative neighbouring criterion

1 Upon receiving a NEIGHBOUR_ACCEPT message from a potential relative neighbour S', the receiving sensor S verifies if there is already a graph neighbour in the Lune created by intercepting the discs with centers in S and S' and radius $d(S, S')$ If such a neighbour exists, then sensor S sends a NEIGHBOUR_DENY message to S'

2 If no existing graph neighbour is found in the previous step, this means that sensor S' is in fact a relative neighbour However, some of the existing graph neighbours could be affected by this newly accepted sensor and they are no longer relative neighbours If the newly accepted sensor S' falls in the Lune between sensor S and one of the existing graph neighbours S_i, the neighbour in question should be excluded by sending it a NEIGHBOUR_DENY message

A description of the main interactions required for the construction of the CDGG and CDRNG are summarized by Algorithm 8 and 9 respectively

Lemma 5. *The swapping-based mobility strategies built on the CDGG guarantee that all sensors reach the rendezvous location within a finite number of swapping operations*

Algorithm 8 GDGG Construction. sensor S and facility F

(* In State $INIT$: *)
send $NEIGHBOUR\_REQUEST$ broadcast message
become $BATTERY\_OK$
(* In State $BATTERY\_OK$: *)
if receiving $NEIGHBOUR\_REQUEST$ from S' **then**
  if $distance(S,F) < distance(S',F)$ and $distance(S,S') < distance(S',F)$ and $DistancePointToLineIn(S,S',F,distanceToLine)$ **then**
    $parentList.Add(S')$
    send $NEIGHBOUR\_ACCEPT$ to S'
  end if
end if
if receiving $NEIGHBOUR\_ACCEPT$ from S' **then**
  $midPoint.X = (S.CoordX + S'.CoordX)/2$
  $midPoint.Y = (S.CoordY + S'.CoordY)/2$
  while $i \leq numNeighbours$ **do**
    if $S.distance(midPoint) \geq neighbourPositions[i].distance(midPoint)$ **then**
      send $NEIGHBOUR\_DENY$ to S'
      become $BATTERY\_OK$
    end if
  end while
  while $i \leq numNeighbours$ **do**
    $midPoint.x = (S.CoordX + neighbourPositions[i].CoordX)/2$
    $midPoint.y = (S.CoordY + neighbourPositions[i].CoordY)/2$
    if $S.distance(midPoint) \geq S'.distance(midPoint)$ **then**
      send $NEIGHBOUR\_DENY$ to $neighbour[i]$
      $neighbourList.Remove(i)$
    end if
  end while
  $rankingPar = d(S,S')$
  $neighbourList.Add(S', rankingPar)$
  $neighbourList.rank()$
end if

Algorithm 9 GDRNG Construction sensor S and facility F

(* In State $INIT$ *)
send $NEIGHBOUR\_REQUEST$ broadcast message
become $BATTERY\_OK$
(* In State $BATTERY\_OK$ *)
if receiving $NEIGHBOUR\_REQUEST$ from S' **then**
  if $distance(S,F) < distance(S',F)$ and $distance(S,S') < distance(S',F)$ and $DistancePointToLineIn(S,S',F,distanceToLine)$ **then**
    $parentList\ Add(S')$
    send $NEIGHBOUR\ ACCEPT$ to S'
  end if
end if
if receiving $NEIGHBOUR\_ACCEPT$ from S' **then**
  while $i \leq numNeighbours$ **do**
    if $S\ distance(S') \geq S\ distance(neighbourPositions[i])$ and $S\ distance(S') \geq S'\ distance(neighbourPositions[i])$ **then**
      send $NEIGHBOUR\_DENY$ to S'
      become $BATTERY\_OK$
    end if
  end while
  while $i \leq numNeighbours$ **do**
    if $S\ distance(S') \leq S\ distance(neighbourPositions[i])$ and $S'\ distance(neighbourPositions[i]) \leq S\ distance(neighbourPositions[i])$ **then**
      send $NEIGHBOUR\_DENY$ to $neighbour[i]$
      $neighbourList\ Remove(i)$
    end if
  end while
  $rankingPar = d(S,S')$
  $neighbourList\ Add(S', rankingPar)$
  $neighbourList\ rank()$
end if

Proof. The proof of this lemma is trivial. Let $G = (V, E)$ be a CDG with vertices $V = \{s_1, ..., s_N, F\}$ where s_i, $1 \leq i \leq N$ represent sensors and F denotes the rendezvous location (i.e., recharge station). Let E be a set of edges of the form $s_i \rightarrow s_j$ where s_j is graph neighbour of s_i. Let $G' = (V', E')$ be a CDGG where $G' \subset$G.
Without loss of generality, we can assume that for any path $P_i =< s_i, ..., s_K, F >\in V'$ leading to the recharge station F, with $1 \leq i < K \leq N$, then $P_i \in V$. From Lemma 4 it is known that P_i does not contain any cycles. Hence, the Lemma holds. □

5.3 Increasing Sensor Knowledge

Another possible strategy to improve the overall performance of the proactive strategy and help low energy sensors reach the recharge station faster is to add additional information about the energy levels of the 2-hop graph neighbours. Regardless of the topology chosen (i.e., CDG, CDGG, or CDRNG), having the 2-hop neighbouring information combined with the 1-hop greedy strategy should lead to a more energy efficient path selection. To implement this new approach, a series of changes to the existing algorithms is necessary. For example, the neighbouring information stored by each sensor s needs to change to include the tuple $(s_i, E_{S_i}, E_{S_{i_{2hop}}})$ where s_i is the *i-th* 1-hop neighbour of s. E_{S_i} represents the energy level and $E_{S_{i_{2hop}}}$ represents the average energy levels of the 1-hop graph neighbours of s_i.

The information about existing 1-hop graph neighbours will be appended to the NEIGHBOUR_ACCEPT messages sent during the graph creation phase. When a sensor sends a NEIGHBOUR_ACCEPT message to its parent, the message will now include the average energy level of its existing 1-hop neighbours. This new piece of information will have to be updated once the migration or swapping phase is initiated. Consequently, two swapping sensors will exchange this new piece of information as part of the swapping process. Furthermore, sensors reacting to a SWAP_COMPLETE

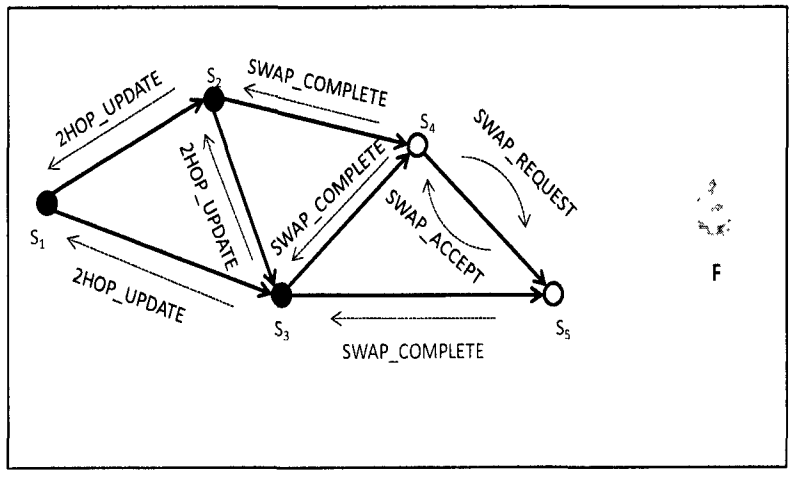


Figure 5.3: Sensor swapping with 2-hop neighbours updates.

message will generate a new message NEIGHBOUR_2HOP_UPDATE to inform their parents about the changes of their 2-hop graph neighbours.

Let us examine the example shown in Figure 5.3 to illustrate the new interactions required during a swapping operation. In this example, sensors S_4 and S_5 have agreed to swap positions after the corresponding exchange of SWAP_REQUEST and SWAP_ACCEPT messages. Once the sensors arrive at the location occupied by their swapping partners, both sensors (i.e., S_4 and S_5) will send SWAP_COMPLETE messages to their parents S_2 and S_3. The SWAP_COMPLETE message received by sensor S_2 contains the tuple $(S_4, E_{S_4}, E_{S_{4_{2hop}}})$. After updating its neighbouring information with the newly received information, S_2 computes the combined energy level of its 1-hop graph neighbours: $E_{S_{2_{2hop}}} = \frac{E_{S_3}+E_{S_4}}{2}$ and sends a new NEIGHBOUR_2HOP_UPDATE$(S_2, E_{S_4}, E_{S_{2_{2hop}}})$ message to its parent S_1.

It is clear from the previous example that for each successful swapping operation there will be an overhead produced by the new NEIGHBOUR_2HOP_UPDATE messages. The density of the graph, determined by the neighbour selection criteria and

the sensor transmission ranges, will have a great impact on how many of these new notification messages are generated. The next section examines the impact of this added knowledge, its relationship with the underlying topology chosen, its potential benefits and possible drawbacks.

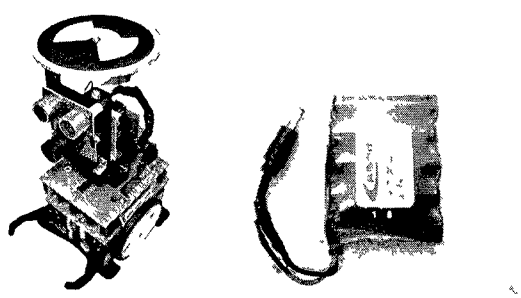
5.4 Experimental Results

Previous work on energy consumption of wireless sensor networks and protocols such as 802.11, have found that the energy required to initiate communication is not negligible. In particular, loss of energy due to retransmissions, collisions and acknowledgments is significant [21, 22]. Therefore, protocols that rely on periodic probe messages and acknowledgments are considered high cost. For these reasons, the design of our algorithm and related coordination had to be flexible enough to avoid the use of probe messages and complicated state-full protocols. It is also noted in the literature that sensors' energy consumption in an idle state can be as large as the energy used when receiving data [22]. On the other hand, the energy used in transmitting data is between 30-50% more than the energy needed to receive a packet. These differences, between the energy needed to perform the basic operations and the percentage of battery usage, vary depending on the communication protocols, hardware and type of battery used.

A common consideration for any solution involving mobile entities is how to accurately represent the cost of energy spent when moving from one location to another. Locomotion cost depends on many factors such as the weight of the electronic components, irregularities in the terrain, obstacles, etc. For simplicity, in [41], the weighted Euclidian distance between origin and destination is used as the cost of relocating a

robot. In this chapter we consider an experimental setting based on real robots deployed in a controlled environment. The goal of the experiments was to identify the impact of basic operations (i.e., communication, sensing, locomotion, idle operation) on the overall battery life.

The experiments were based on the PropBot 2.0 mobile robot (Figure 5.4(a)) developed in the School of Computer Science Robotics Lab at Carleton University. The robot's hardware specifications include: Parallax Propeller Microprocessor (with 8 processors), Parallax Continuous Rotation Servos, CUMCam camera, three Sharp GP2Y0D810Z0F Digital Distance Sensors and Nubotics WW-01 Encoders. Communications use the Parallax EasyBluetooth module and batteries are custom-made 6v battery packs using 2600mAh NIMH AA cells (Figure 5.4(b)). A single mobile robot was deployed in an area of 2m x 1.5m and tests were performed to determine battery drain under the following conditions: 1) idle state, 2) continuous movement, 3) communication 4) sensor usage.



(a) PropBot 2 0 (b) Battery Pack

Figure 5.4: Experimental equipment

Our preliminary results show that energy spent in communications (i.e., send/receive) is 25% more than the battery drain in the idle state. Battery drain under perpetual movement (i.e., locomotion costs) is almost twice as much as communication cost.

Sensing with the CUMCam was among the most costly operations and the battery recharge was 14x faster than battery drain in the idle state. These findings were incorporated into our simulation scenarios to study the impact of critical variables on the overall solution.

The simulation scenarios are implemented in Omnet++ [77] along with the mobility framework extension [19]. For all experiments, the sensors and charging facilities were randomly placed in an area of 1000x1000m^2. The analysis of our simulated results centers on three important aspects of the solutions:
1) Whether or not a state of equilibrium is achieved and the number of sensor losses until such condition is met.
2) Impact of several variables such as: underlying topology, transmission range, mobility cost and sensor knowledge.
3) Role played by the charging station: passive vs. active.

In all cases, the quality of the strategy is measured in terms of optimal runs vs. panic runs. Constant cost values are assigned to each basic operation (i.e., send, receive, idle and move). Initial values for these operations are based on the observations with the PropBot robot as well as some of the experiences found in the literature [21, 22].

5.4.1 Transmission Range and Mobility Cost

This experiment was designed to verify the impact of the sensor's transmission range on the overall performance. The characteristics of the network were the same as the test performed in Chapter 4 involving 100 sensors and one service facility. The facility is equipped with two sockets, which allow only two sensors to be recharged at the

same time. A series of 30 experiments with different random deployments were run for 10^6 simulation seconds. The energy ratio for sending/receiving a packet is set to a constant ($E : E/2$). Locomotion costs were based on the weighted Euclidean distance with a weight factor of $1/5E$ per meter traveled. The only difference is that the transmission range was varied from 50m, 75m, 100m, 200m, 300m and 400m.

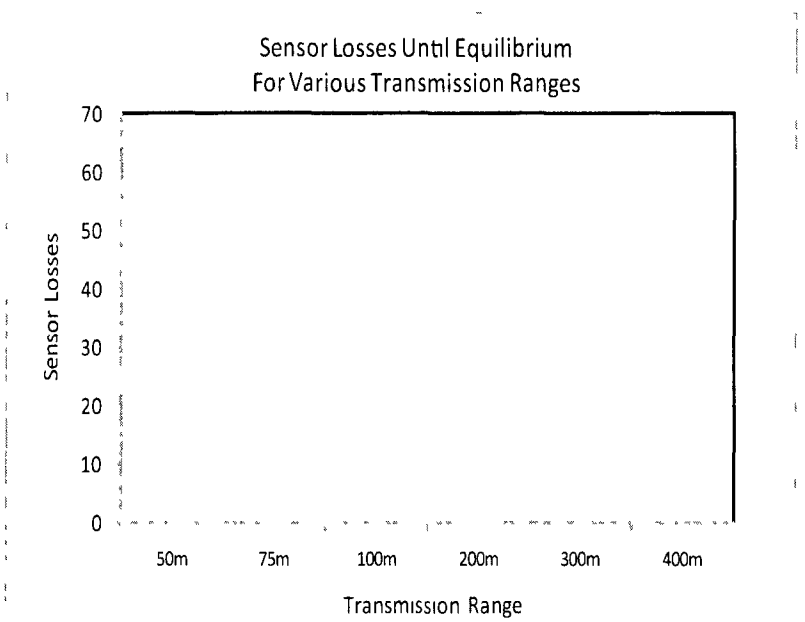


Figure 5.5: Sessor losses over time - variable range.

Figure 5.5 shows the cumulative number of sensor losses until equilibrium for each range value. In a deployment of 1000x1000m^2 a transmission range of 50m was too restrictive, which means that most of the sensors were isolated and the number of immediate neighbours in the CDG was too small to guarantee a gradual approach towards the recharge location Another interesting observation is that by increasing the transmission range, the number of losses decreased dramatically. However, for larger ranges (e.g., 300m and 400m), there was a decline on the overall performance since many neighbours are discovered, resulting in an added overhead to maintain more

information per sensor as well as additional interactions due to update messages as a result of successful swapping and recharging operations.

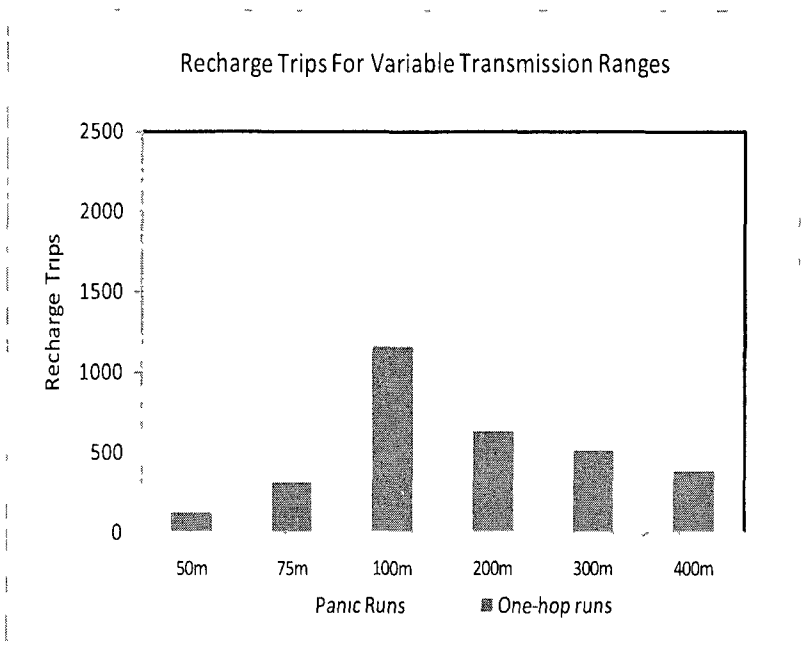


Figure 5.6: One-hop runs vs. panic runs - variable range.

Figure 5.6 shows the quality of the solution in terms of one-hop runs vs. panic runs. In an ideal system, our solution should reach the state of equilibrium using one-hop runs only. As expected, for a transmission range of 50m, most of the trips could be considered panic runs since there is almost no migration due to the lack of 1-hop neighbours. The best breakdown between one-hop and panic runs occurs with 100m range. However, there are more visits to the recharge location, when compared to the 200m, 300m and 400m cases. Although there is no clear explanation for this phenomenon, one can argue that there is a trade-off between the total number of recharge trips and the breakdown between one-hop vs. panic runs. In a panic run situation, a sensor travels from a more distant location and after having been recharged, it needs to return farther to its initial location. This situation creates a

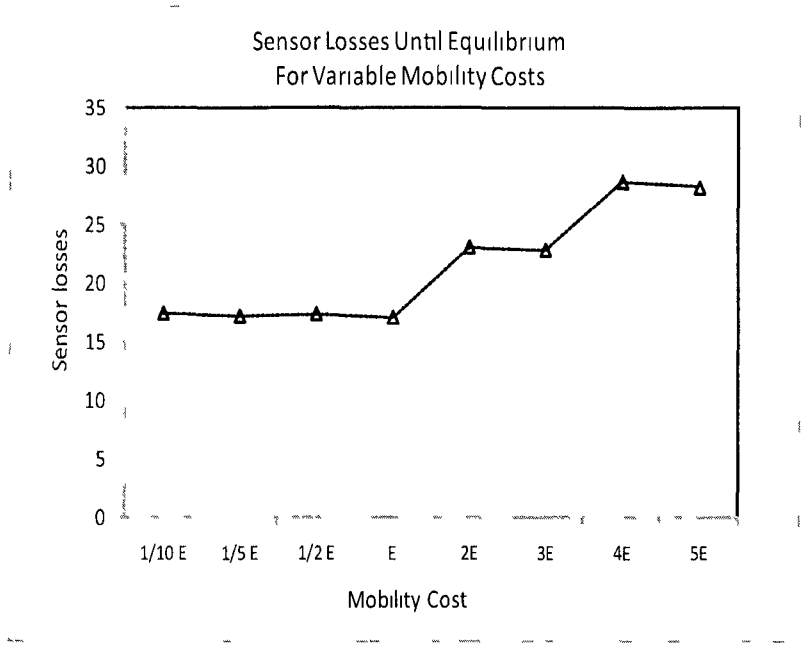


Figure 5 7 Sessor losses over time - mobility cost

coverage hole that lasts longer than holes created by one-hop runs However, more one-hop recharge trips also means more coverage holes but for shorter periods of time

The next experiment explored the impact of the locomotion cost on the number of losses until equilibrium was reached as well as the distribution and number of recharge trips The network setup remained the same with the transmission range fixed at 100m Figure 5 7 shows the number of losses until equilibrium for several mobility costs The cost function is based on the weighted distance traveled by the sensors, with the weight constant w defined as a function of the energy spent to send a packet For example let E be the energy spent to send a packet over the 100m range, then for each meter traveled, the sensor will spend wE units of energy, where $w \in \{1/10, 1/5, 1/2, 1, 2, 3, 5\}$ In other words, the energy spent to move the robot 100m, ranges from 10x to 500x the energy required to send a packet over the same distance In particular, the values observed for the ProbBot robot fluctuated around

54x the communication energy.

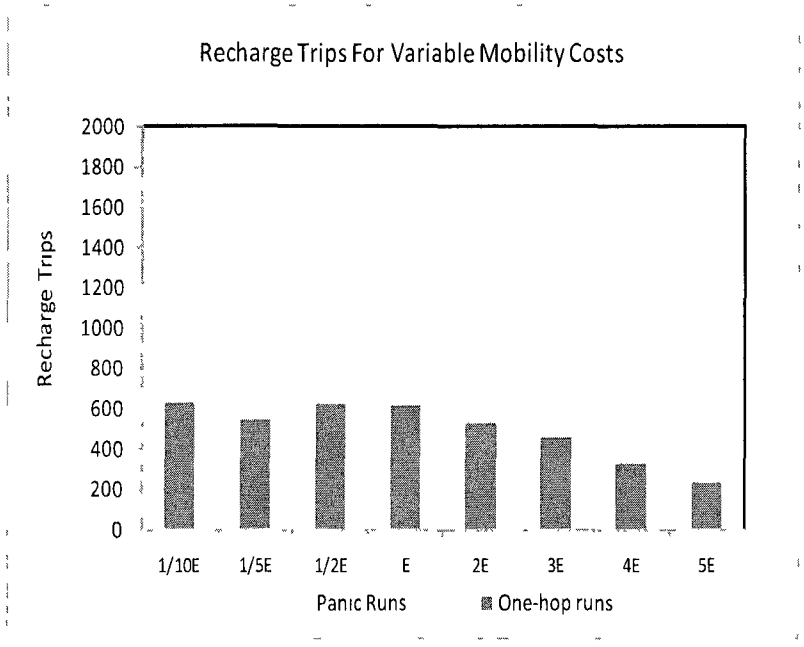


Figure 5.8: One-hop runs vs. panic runs - mobility cost.

The simulation results show that as the locomotion costs increase (in relation to the transmission cost) so does the number of sensor losses until equilibrium. The trend seems to be closer to a step function with clear discrete increments at some values. Another observation is that despite the increase in the number of sensor losses, the network survivability is still over 70%, even for the worst case.

In terms of the quality of the solution in the variable mobility cost scenario, Figure 5.8 shows the same step function behaviour for the total number visits to the station. However, there is a significant degradation on the number of one-hop trips as the locomotion cost increases because a larger number of sensors fall into the BATTERY_CRITICAL state before completing their migration.

5.4.2 Topology Comparison

This test was designed to determine whether our proactive solution to FFP reaches a state of equilibrium when the new proposed CDGG and CDRNG are used as the underlying topologies for the mobility strategies. The experiment measured the cumulative number of sensor losses until energy equilibrium is reached.

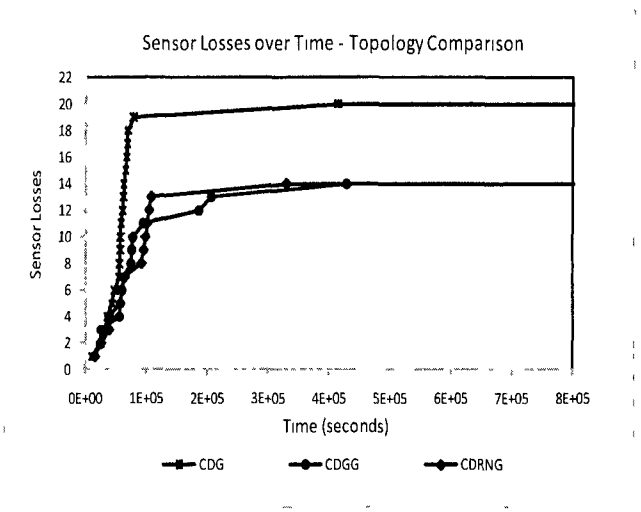


Figure 5.9: Sensor losses until equilibrium. Topology comparison.

Figure 5.9 shows the result of a simulation involving 100 sensors and one service facility. The facility is equipped with two sockets, which allow only two sensors to be recharged at the same time. A series of 30 experiments with different random deployments were run for 10^6 simulation seconds. The sensor transmission range is now fixed at 100m and the energy ratio for sending/receiving a packet is set to a constant $(E : E/2)$. Locomotion costs were based on the weighted Euclidean distance with a weight factor of $\frac{1}{5}E$ per meter traveled. For all the tests performed on the three different topologies, the mobility strategy selected was the greedy closest-first swapping where a low energy sensor chooses its closest graph neighbour as a swapping partner during its migration towards the recharge station.

As expected, the closest-first swapping strategy on the three topologies chosen (i.e., CDG, CDGG and CDRNG) reached the state of equilibrium. The CDGG and CDRNG are sub-graphs of the CDG and according to the experimental results presented in Chapter 4, even the single path (i.e., single neighbour) approach reached the state of equilibrium. However, the interesting finding is that although the three topologies reached the state of equilibrium at the same time approximately, the CDGG and CDRNG reported fewer sensor losses due to battery depletion. This is an important observation that implies that fewer but better selected graph neighbours will yield better results if the main goal is to minimize the number of permanent failures due to battery depletion.

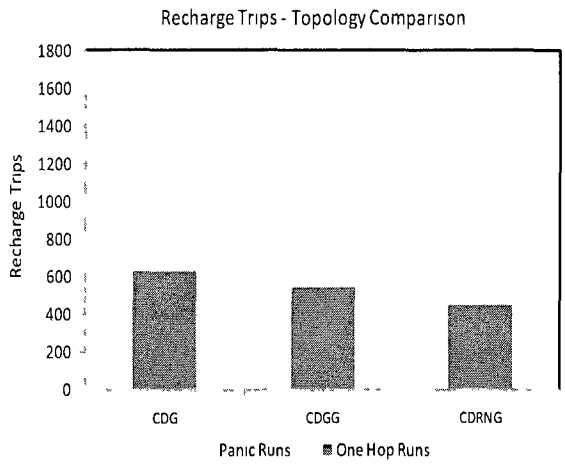


Figure 5.10: Number fo recharge trips. Topology comparison.

Unfortunately, the CDGG and CDRNG did not report any improvements in terms of optimal trips to the recharge station. Figure 5.10 shows the number of recharge trips and breakdown between optimal and panic runs for the three topologies in question. For the CDGG and CDRNG there was a small increase in the number of

recharge visits compared to the CDG and a small decrease in the number of optimal runs This decrease is somehow expected since the number of neighbours for both topologies (i e , CDGG and CDRNG) is more restrictive than the CDG. Once more, choosing different topologies for the migration strategy exposed a trade-off between permanent coverage holes due to battery depletion and more short-lived temporary holes due to more frequent visits to the facility

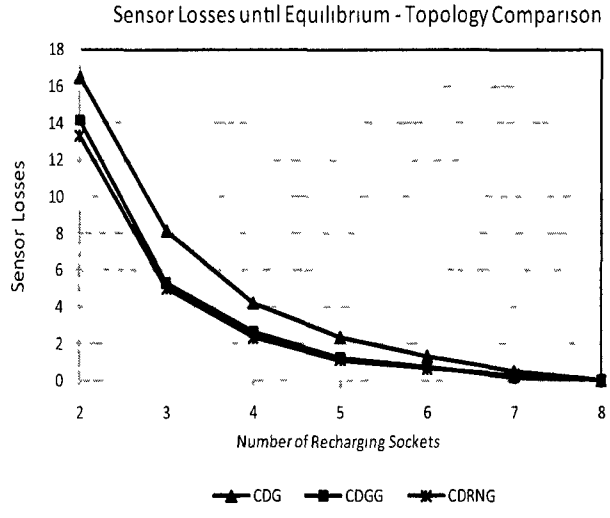


Figure 5.11. Sensor losses until equilibrium by number of recharge sockets Topology comparison.

The next part of this test was designed to measure the impact of the recharge sockets on the cumulative number of losses until equilibrium and verify whether the perfect equilibrium can be reached by increasing the number of sockets or docking ports in the recharge station. The network setup remained the same and the closest-first greedy mobility strategy was tested on the three topologies (i.e., CDG, CDGG and CDRNG). Figure 5.11 shows the result for this test where the closest-first swapping strategy on the three topologies showed the same progression towards perfect equilibrium The total number of recharge sockets needed for the perfect equilibrium

is the same for the three topologies but the CDGG and CDRNG showed an improvement on the number of sensor losses over the CDG as the number of recharge sockets increased.

5.4.3 Sensor Knowledge

The goal of this set of tests is to verify the impact of added sensor knowledge, as introduced in 5.3, and compare it with the 1-hop information greedy strategies on the three proposed topologies. The network parameters are the same as in the previous tests, with fixed transmission range at 100m. The closest-first swapping strategy is applied on the three topologies (i.e., CDG, CDGG and CDRNG) with information about the energy levels of 1-hop graph neighbours only and 2-hop graph neighbours respectively.

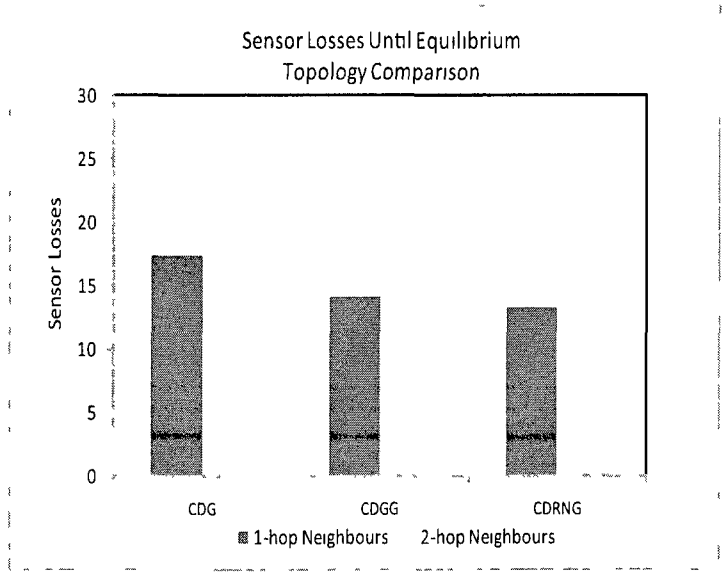


Figure 5.12: Sensor losses until equilibrium. Topology comparison.

Figure 5.12 shows the number of sensor losses until equilibrium for the three topologies tested with 1-hop neighbour information vs. 2-hop neighbour information.

In each case, there was an increase in the number of sensor losses when the migration strategy included the 1-hop neighbour information. When 2-hop information is used, the best performer was the CDGG with losses similar to the 1-hop CDG. This is a rather surprising result, which seems to imply that "knowing more individually" about the network is less useful for the collective effort than "knowing less". Knowing more in this case has a direct impact on the number of control messages required to maintain the underlying topology in a consistent state. This phenomenon will be more evident as the graph degree increases. The graph maintenance overhead related to keeping 2-hop neighbour information proved to be crucial to the point that counteracts any possible improvement when compared to keeping 1-hop information only.

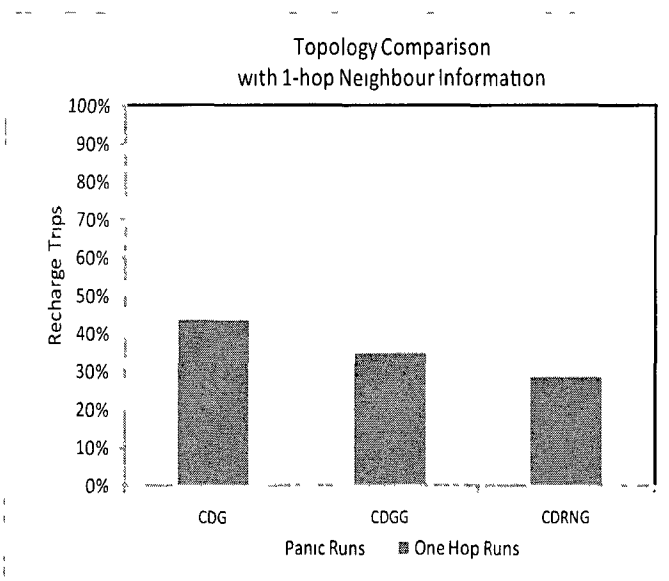


Figure 5.13: Recharge trips with 1-hop neighbour information. Topology Comparison

The idea of adding extra knowledge to the sensors aimed to improve the path selection strategy and increase the number of optimal runs or 1-hop trips to the recharge station. The simulation results shown in 5.14 confirmed our expectations.

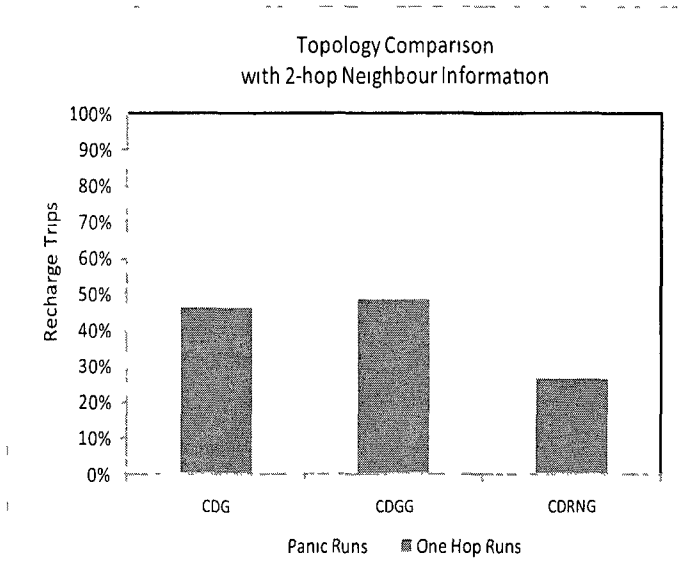


Figure 5.14: Recharge trips with 2-hop neighbour information. Topology Comparison

Added knowledge had, in fact, a positive impact on the selection of the better energy-efficient migration strategy towards the recharge station. There was some marginal improvement on the number of optimal runs for the CDG and CDRNG with a real improvement for the CDGG. The CDGG proved again to be the best performing topology in terms of cumulative sensor losses until equilibrium and breakdown between panic and optimal runs when using 2-hop neighbour information.

The last test involving the added-knowledge scenario examined the impact of the sensors transmission range on the overall performance. For this test, the closest-first swapping strategy on the CDG with 2-hop neighbour information was implemented on the network of 100:1 sensor/facility ratio with various transmission ranges (e.g., 50m, 100, 200m, 300m).

Figure 5.15 shows the cumulative number of sensor losses until equilibrium for each range value. The behaviour is very similar to the 1-hop neighbour information

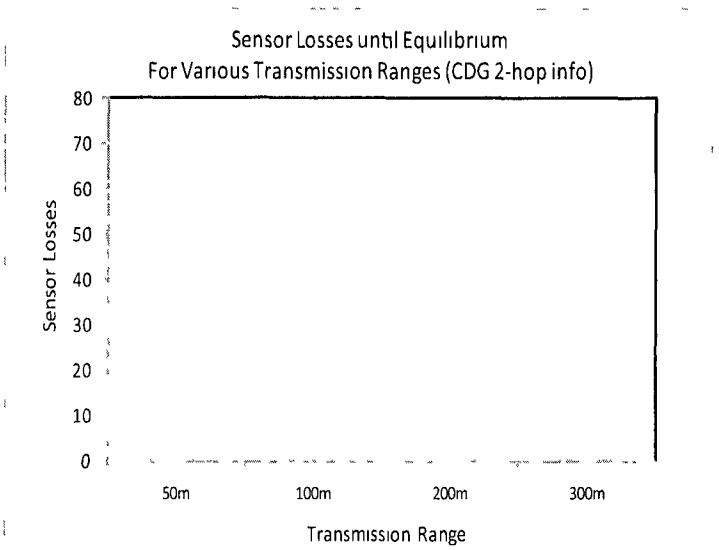


Figure 5.15: Sensor losses until equilibrium with 2-hop neighbour information. Various transmission ranges.

scenario shown in Figure 5.5. The transmission range of 50m was too restrictive, which means that most of the sensors were isolated and the number of 1-hop and 2-hop neighbours in the CDG was too small to guarantee a gradual approach towards the recharge station. By increasing the transmission range, the number of losses decreased dramatically. However, for the 300m range there was a decline on the overall performance, which is consistent with the 1-hop information scenario.

The number of recharge trips and breakdown between panic and optimal runs is shown in Figure 5.16. Following the same behaviour as in the 1-hop information scenario, for a transmission range of 50m, most of the trips could be considered panic runs since there is almost no migration due to the lack of 1-hop neighbours. The best breakdown between one-hop and panic runs occurs with the 100m range. However, there are more visits to the recharge location, when compared to the 200m and 300m cases, which reported more balanced results in terms of the number and type of visits to the facility.

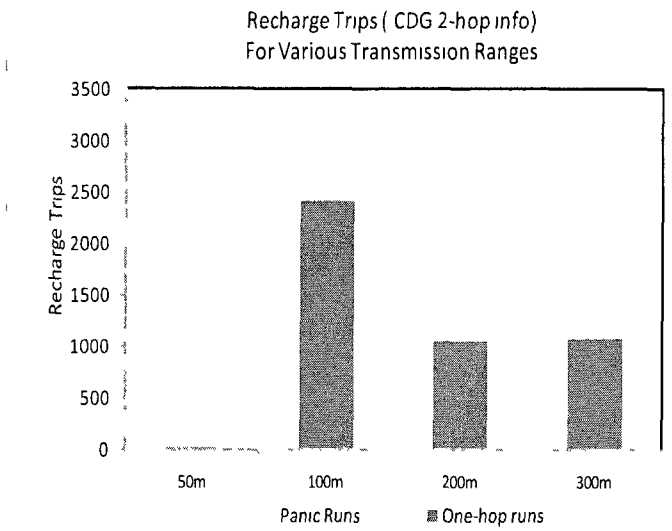


Figure 5.16: Number of recharge trips with 2-hop neighbour information. Various transmission ranges.

5.4.4 Passive vs. Active Recharge Station

During the algorithm, the facility plays a rather passive role. The facility's responsibilities are limited to keeping a queue of waiting sensors ranked by their energy levels and notifying the sensors when a socket or docking port becomes available. In a passive scenario, a socket becomes available when the sensor has reached 100% of its battery level and sends a RECHARGE_DONE message to the facility. In an active scenario, as depicted in Figure 5.17, the facility does not have to wait for the sensor's battery to be 100% recharged. In this case, a sensor will notify the facility when its battery has reached an operational level (e.g., 75%). Consequently, the facility could halt the charging process by sending a TERMINATE_RECHARGE, if there are other sensors waiting in line.

The final experiment examined the case where the recharge station was given a

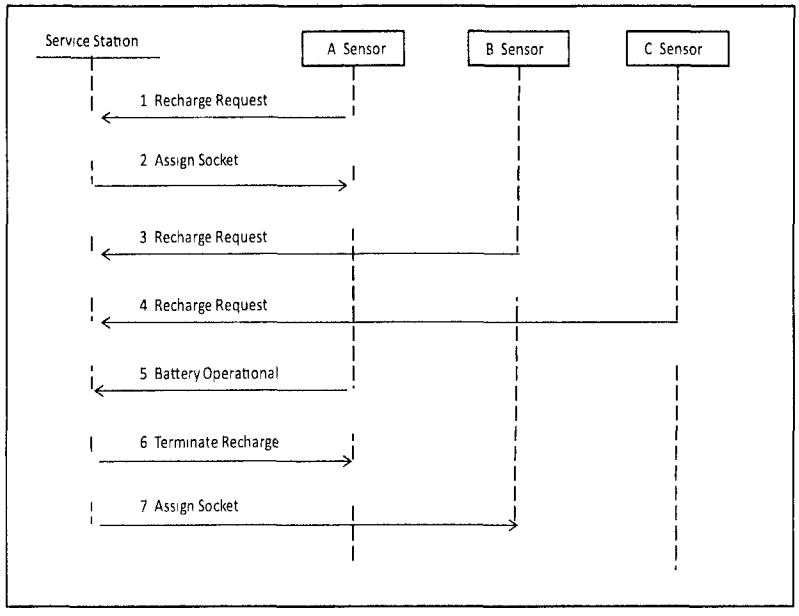


Figure 5.17: Active station pattern.

more active role in order to minimize sensors losses while waiting for an available socket. Figure 5.18 shows the comparison between passive vs. active charging stations while using the same sensor deployment as the previous tests. These particular tests used a fixed transmission range of 100m, a mobility cost factor of 1/5E per meter traveled and a recharge rate 10x faster than battery drain in idle state. In this experiment, the sensors notified the station when their batteries reached 100%, 75% and 50% of charge.

By giving the facility a more active role, the network survivability at the state of equilibrium improved from 80% to close to 90% for the 75% recharge case. The number of one-hop trips also improved from 37% to 41% for the 75% sensor recharge case. However, the number of recharge trips increased by 30%. Recharging the batteries to 50% of their capacity almost doubled the number of recharge visits when compared to the 75% case. The number of one-hop trips also decreased but the network survivability remained close to 90%. Once again there is a trade-off between creating

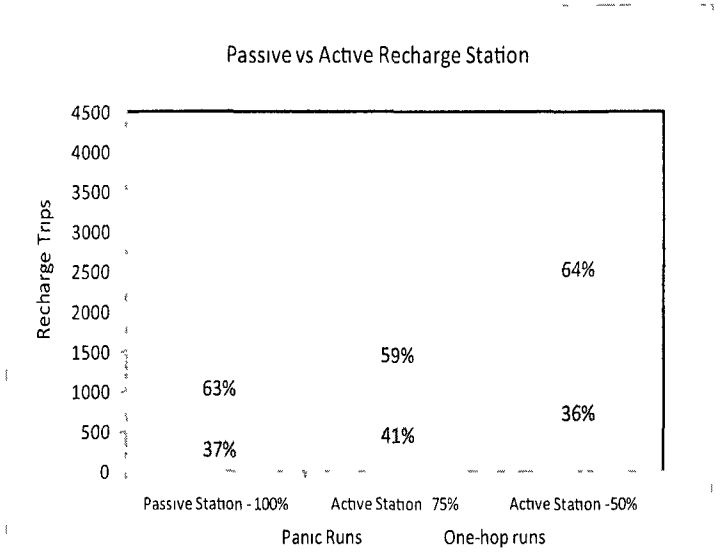


Figure 5 18: Passive vs Active Recharge Station.

temporary coverage holes produced by additional recharge trips and permanent coverage holes produced by sensor losses due to battery depletion.

5.5 Conclusions and Summary of the Experimental Results

In this chapter, we have enhanced the proposed proactive strategies to solve the facility absorbed Frugal Feeding Problem (FFP) introduced in Chapter 4. In summary the proposed modifications are:

1. Introduction of new underlying topologies with different neighbour selection processes (e.g., Compass Directed Gabriel Graph and Compass Directed Relative Neighbor Graph). The proposed graphs guarantee that sensors will reach the rendezvous location within a finite number of swapping operations with a loop-free migration trajectory. The proposed CDGG, CDRNG are dynamic and self-correcting: neighbouring information is updated following every successful swapping or recharge operation.

2 Enhance sensor capabilities and decision making by adding information about energy levels of the 2-hop graph neighbours All decisions made by the sensors regarding the next swapping operation are based on local knowledge (i e , the algorithms are completely distributed and localized) However, a new look-ahead parameter that includes the combined energy levels of the 2-hop neighbours is taken into account in the selection of the swapping partner

3 Enhance the recharge facility role by adding a more active behaviour The facility will now monitor the recharging sensors as well as its queue of waiting low-energy sensors and halt the recharge process for those sensors that have reached an operational level

The experimental analysis of the modified proactive solution to the FFP shows that

1 For networks of 100 1 sensor/facility ratio, the network survivability rate can be improved by using a CDGG or CDRNG as an underlying topology for the migration strategy

2 Having more individual knowledge does not necessarily help in the collective effort Adding the energy levels of the 2-hop graph neighbours improves a sensor's individual migration strategy towards the facility There is an increase in the number of optimal trips However, the number of losses until equilibrium also increases, which results in lower network survivability

3 If 2-hop neighbour information is available, the proposed CDGG outperforms the other proposed topologies in terms of network survivability at the point of equilibrium and distance traveled to the facility

4 The transmission range has a positive impact on the network survivability at the point of equilibrium and the number of optimal trips to the facility However, for higher transmission ranges that result in higher degree graphs, there is a

clear negative impact on the key quality indicators (i e , sensor losses, optimal trips, total number of recharge trips)

5 By giving the facility a more active role during the recharging process, the state of equilibrium can be reached with close to 90% network survivability The breakdown between optimal and panic runs is also improved but there is an increase in the overall number of recharge trips

6 In general, the simulations exposed several trade-offs between the key variables (i e , topology, transmission range, locomotion cost, sensor knowledge and station role)

Chapter 6

Conclusions, Extensions and Open Problems

6.1 Conclusions

Throughout this thesis we have explored different alternatives to extend the operating life of Wireless Sensor Networks The main goal was to increase network availability by recharging, replacing or redeploying "depleted" sensors with the help of mobile entities The problem domain was divided into two main categories 1) networks of static sensors with mobile maintenance facilities/robots and 2) networks of mobile capable sensors and static maintenance facilities

For the static sensor scenario, we proposed a balanced cluster-based energy management system combining a static cluster partitioning with a distance-aware robot positioning The maintenance robots should coordinate their work by partitioning the network into balanced clusters and positioning themselves at the center of these clusters This work has also shown that the construction of such balanced partition in an asynchronous environment is possible if the robots are provided with some memory capabilities

The experimental analysis of our proposed clustering algorithms showed that networks with higher sensor-robot ratio performed better than those with a lower ratio in terms of the average number of messages per sensors and robots Although there were differences in the communication cost, the algorithms produced a cluster partition with similar quality (i e , similar average sensor distance to cluster centers and

sum of the distances) All the proposed clustering solutions produced partitions with similar quality when compared to the centralized benchmarks and they all showed a fast progression towards convergence This translates into rapid improvement in the initial rounds with minor adjustments thereafter

We also showed that cluster-based management solutions can be extended to provide a distributed and balanced solution to the Facility Location Problem (FLP) The FLP attempts to find suitable locations for service facilities subject to a particular optimization criterion For this kind of environment, we provided a cluster-based solution where (1) workload is balanced, (2) the movement of the maintenance entities in their partition is minimized, and (3) the number of sensor communications is small While finding the optimal partition and placement of the facilities is a NP-hard problem, we showed a simple and efficient solution that produced partitions of remarkable quality Our proposed solution focuses on the capacitated version of the FLP where the facilities can only serve a predefined number of clients, they all have to be deployed and there is no restriction on the eligible locations for deployment Our approach combined an initial cluster partitioning, possibly balanced, with an optimization phase to find the optimal placement of the facilities The final placement satisfied the condition of load balance among all facilities as well as minimized the total facility travel time The clusters are optimized through a sequence of iterations or "virtual swapping" of sensors, each time creating a partition whose quality is strictly better than the previous In particular, in each iteration of the algorithm, sensors are virtually swapped (i e , their cluster membership changes) if, and only if, by doing so the quality of the overall solution improves Convergence is achieved in a finite number of steps At this point, cluster membership becomes static and the facility deployment is final

Our sensor swapping FLP approach is completely distributed There is no central entity with global knowledge and the virtual swapping operations take place only between adjacent areas Sensors only need to know about their assigned maintenance facility Experimental analysis showed that the cost in terms of sensor messages remains constant as the network size increases The quality progression towards convergence and the quality of the final deployment are similar to the selected centralized benchmarks (i e , K-Means and HAC)

The second part of this thesis examined the problem of energy management and restoration in a mobile network scenario In particular, the focus was on networks with mobile sensors and static service facilities, where the sensors have to rendezvous with the facilities to recharge their batteries This problem is known as the facility absorbed Frugal Feeding Problem (FFP) However, the facility resources (i e , recharging sockets) are limited and sensors should coordinate their actions to access these shared resources efficiently Existing energy management approaches are mainly reactive and based on fixed threshold as the deciding factor in the strategy to follow Under these circumstances, we posed the question Should the sensors wait until their batteries fall below the predefined thresholds or should they take a more proactive approach while they are fully operational? This work provided some answers to these issues by comparing passive vs proactive approaches to energy management based on different mobility strategies

Our solution to the facility absorbed FFP recommends taking a proactive approach to energy restoration based on several mobility strategies As the foundation for their mobility strategies, sensors create a logical Compass Directed Graph (CDG)

In particular, we proposed to reduce the problem of coordinating the recharge of mobile sensors to finding optimal routes in a CDG built on top of the original deployment The proposed graph incorporates ideas from forward progress routing and the directionality of compass routing in an energy-aware unit graph The idea behind each mobility strategy is that sensors will swap positions with graph neighbours with higher energy levels in order to get closer to the service station The mobility modes are built upon routing concepts but instead of sending packets, sensors navigate on the logical graph until they reach the rendezvous location

The proposed graph topology guarantees that any node reaches the service facilities in a finite number of swapping operations (i e , the trajectory is loop-free) All decisions made by the sensors regarding the next swapping operation are based on local knowledge (i e , the algorithms are completely distributed and localized) The proposed underlying topology is position-based Sensors create a partial network map with the coordinates of their immediate neighbours and parents The proposed graph is dynamic and self-correcting, new sensors can be added or removed at any time and new neighbours are re-discovered any time a successful swapping or recharge operation takes place

The experimental analysis showed that all the variations of the proactive approach (e g , closest-first, variable degree, closest-with-most-energy) outperformed the threshold based passive approach by a significant margin The closest-first proactive strategy yielded the best results among all proactive solutions, reaching a state of equilibrium with fewer sensor losses The closest-first strategy also provided the most balanced solution, where 40% of the recharging trips were initiated from a one-hop distance to the service station For all the proposed strategies, the state of perfect equilibrium (i e no losses reported) could be reached by increasing the number of

recharging sockets assigned to the facilities However, the passive solution needed twice as many sockets when compared to the closest-first active strategy

In general, the simulations exposed several trade-offs between the key variables (i e , topology, transmission range, locomotion cost, sensor knowledge and station role) For example for networks of 100 1 sensor/facility ratio, the network survivability rate can be improved by using a CDGG or CDRNG as the underlying topology for the migration strategy Adding the energy levels of the 2-hop graph neighbours improves the breakdown between optimal and panic visits to the facility However, the number of losses until equilibrium also increases, which results in lower network survivability Finally, by giving a more active role to the facility, a state of equilibrium can be reached with close to 90% network survivability The breakdown between optimal and panic runs is also improved but there is an increase in the overall number of recharge trips

6.2 Extensions and Open Problems

In this section we present some ideas to further this research as well as some suggested open problems In particular, the emphasis is placed on the mobile sensor scenario and the use of proactive approaches to energy management In Chapter 4, it is shown that pursuing a proactive approach to solve the problem of recharging mobile sensors is not only feasible but also more efficient (i e , when compared to a passive approach) in terms of the number of facility resources needed and overall number of sensor losses until equilibrium The proposed logical topologies provided a dynamic and robust foundation for sensor mobility and coordination Consequently, new questions arise Is it possible to improve the proposed solutions by using a completely different logical topology? And is it also possible to improve the proposed mobility strategies regardless of the underlying topology chosen? How would a proactive strategy work with the presence of obstacles? The next sessions introduce the discussion around these topics along with some possible directions

6.2.1 Achieving a Stable Partition in Multi-hop Sensor Deployments

This section discusses some modifications to the algorithms presented in Chapter 2 to create a clustering partition in a multi-hop WSN In this scenario, the sensors and robots are randomly deployed creating a connected multi-hop deployment Sensors have limited transmission range R, where $R < R'$ and R' is the robot's transmission range, which is assumed to be large enough to reach any sensor or robot in the network Data exchange between two distant sensors (i e , out of range) is forwarded by other sensors That is, there is a routing infrastructure of geo-casting in place where only sensors in the routing path (e g , greedy geo-casting) will forward the data The communication environment is contention and error free, where no retransmissions

are required unless they are part of the algorithms Sensors are still able to adjust their transmission ranges (R), in this case to reach a target robot However, the use of this feature will be limited to only a few selected sensors

In general, the clustering algorithms presented in Chapter 2 (e g , memory only or sensor selection) will have the same behaviour in the multi-hop sensor scenario However, in this case, the robots will designate a cluster head among the sensors in its cluster for direct communication This cluster head sensor will adjust its transmission range if the robot goes out-of-range after traveling to the cluster center That is, robot r with cluster C_r and center C selects a maintenance cluster head sensor $CH \in C_r$ that satisfies $d(CH, C) = R_{min}$ where $R_{min} = min \{d(s_i, C)\} \forall S_i \in C_r$

Consequently, after completing an iteration of the memory-only or sensor selection algorithm, robot r sends a modified $END\_ROUND(CH, R_{min})$ message Sensors in C_r will use the CH sensor to communicate with robot r if direct communication is not possible Furthermore, if $R_{min} > R$ then the maintenance cluster head CH will adjust its range R to the new value R_{min} While the amount of communication will increase in the multi-hop WSN environment, the quality of the final clustering partition remains unchanged

6.2.2 The FFP with Adaptive Threshold Selection

Another research direction deals with possible enhancements to the strategies presented in Chapter 4 Regardless of the topology chosen, we propose to explore some modifications to the proactive mobility strategies In particular, we propose to enhance the threshold selection process

For the proactive strategies, the choice of threshold may affect the performance and overall behaviour of the system For example, relatively low values for the BATTERY_LOW threshold may result in a late reaction for sensors located at larger distances from the station Higher threshold values on the other hand, may trigger an early reaction from these sensors, resulting in unnecessary swapping operations Furthermore, if BATTERY_LOW threshold values tend to be close to the BATTERY_CRITICAL thresholds, the overall system behaviour resembles that of a passive or reactive approach

An alternative to the fixed threshold values, which are usually based on the remaining battery life, is to evaluate the impact of a more adaptive threshold selection In this scenario, the sensors would take into account the maximum number of hops or swapping operations required to get to the facility when computing the BATTERY_LOW and BATTERY_CRITCAL thresholds These values will change as the sensors migrate towards the facility and should be large enough to accommodate the worst possible route in terms of number of hops and total distance traveled

A more complex threshold-based decision making requires that sensors store some extra information along with the existing routing tables Since all swapping decisions are based on local information, this additional piece would require an extra computational step The process should be initiated after the construction of the logical topology (e g , CDG, CDGG, or CDRNG) is completed and will be started by the "leaf sensors" (i e , sensors with no graph neighbours), located at 1-hop distance to the facility These sensors will send a message to their parents indicating their distance to the facility Intermediate sensors, after receiving information from all their graph neighbours, will compute the maximum number of hops and forward this information to their parents Figure 6 1 shows the interactions required to complete

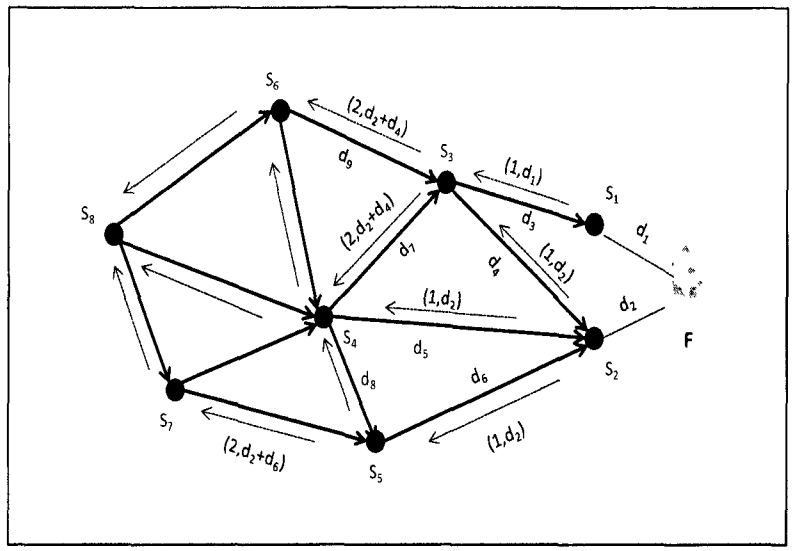


Figure 6.1: Computation of the hop-based thresholds.

this new computational stage.

In general the new interactions can be summarized as follows:

1. If numNeighbours = 0 then send HOP_COUNT(1,distance_to_facility) message to parents.

2. If a sensor receives a HOP_COUNT(num_hops,distance_to_facility) message then it should keep track of the maximum number of hops received so far and the corresponding distance. After receiving HOP_COUNT messages from all its graph neighbours, the sensor will send a HOP_COUNT(max(num_hops)+1, max(distance_to_facility)+d) where d is the distance to the sensor from where the maximum values were received.

3. The process terminates when the sensor does not have any parents.

Consequently, the effectiveness of the new proposed strategies will be affected by this new threshold selection process and should be evaluated through a series of simulations. The experiments should evaluate the time and number of failures needed to

achieve equilibrium, breakdown between one-hop and panic runs and the number of recharging sockets needed to achieve a perfect equilibrium All the new results should be compared with the experiments carried out in Chapter 4 and 5

Future enhancements to this work might also explore the impact of mobility in more detail Locomotion costs are very dependent on physical conditions, hardware specifications, battery technology, etc This may involve the use of the PropBot mobile robots in larger scale implementations of our proactive solution Another possibility may also include the study of other underlying topologies based on a different neighbour selection process

6.2.3 The FFP with Obstacles

The proactive strategies to the facility-absorbed FFP presented in Chapters 4 and 5 assume that there are no obstacles between the sensors and the chosen recharge facility In this section we discuss the scenario where the space may contain static obstacles that prevent the sensors from communicating with other sensors or traveling directly to the recharge station

The presence of obstacles has been the focus of attention in several research papers For example, in [12] a model for obstacles in multi-hop sensor networks is discussed The obstacles are classified in two categories physical obstacles and communication obstacles According to [12], a physical obstacle prevents the deployment of the sensor in that area (i e , a network area which does not contain any sensors) A communication obstacle, on the other hand, causes a disruption to the wireless communication If the line of sight between two sensors crosses the obstacle then there is no communication between the sensors

The problem of how to successfully route packets around obstacles has been examined in [11] In particular, the strategies are divided into passive vs active A passive routing strategy around obstacles does not maintain any information about the obstacle The obstacle avoidance mechanisms are built into the routing algorithm by applying a recovery strategy (e g , [9, 27]) An active approach maintains information about the obstacle shape and size and the sensors will forward the packets using the shortest route towards the destination In general, active strategies to obstacle avoidance create a convex forbidden zone containing the obstacle The forbidden zone is delimited by some marker sensors that prevent the packets from entering a concave area where a simple greedy routing will not work

In our mobile sensor scenario the obstacle avoidance strategies have a higher degree of complexity since the algorithms have to guarantee not only communication but also movement around the obstacles In our scenario, the static obstacles found between the sensors and their closest recharge station can be classified as follows

1 Physical obstacles These obstacles impede the movement of the sensors through the obstacle (e g , walls, ditches, lakes, etc) but there is communication between sensors separated by the obstacle

2 Communication obstacles Sensors in the line of sight cannot communicate but they can physically travel through the obstacle (e g , communication interference, blackout area)

3 Physical and communication obstacles, blocking both movement and communication of sensors

Notice that once a sensor enters a blackout area, communication is lost and if a failure occurs, there is no possible recovery unless the sensor exits the area For this

reason, we can assume that, if there is no communication between a pair of sensors within range (i.e., sensors are separated by a blackout area), neither sensor will enter the area in order to get to the other sensor's location. Typically, sensors have limited vision or capabilities to detect physical obstacles unless they are in close proximity. On the other hand, sensors are always able to circumnavigate around any physical obstacle.

The presence of obstacles affects the creation of the CDG/CDGG needed for the FFP proactive strategies. Consequently, our approach to obstacle avoidance falls into the active category, where the CDG should guarantee that no sensors get trapped in a concave area during their migration towards the recharge station. The sensors should create a convex area around the obstacle and build the directed graph using the bordering sensors that provide progress towards the recharge station. The obstacle detection and avoidance decisions should be based on local information only.

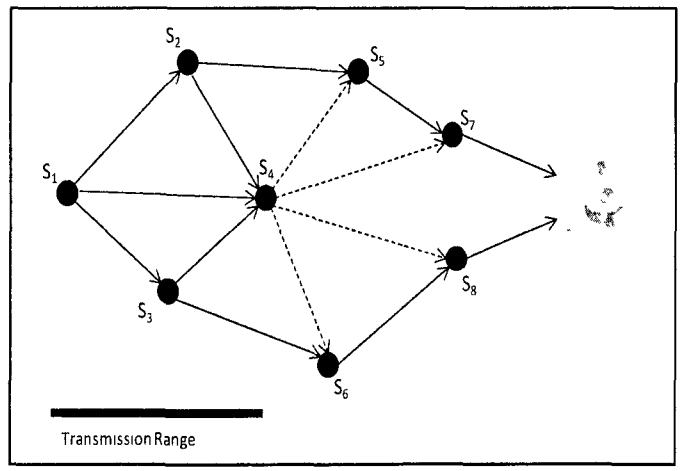


Figure 6.2: CDG construction with a single obstacle.

Figure 6.2 shows a sensor deployment with a single static obstacle. The dashed lines in the CDG represent the edges ($s_4 \rightarrow s_5, s_4 \rightarrow s_6, s_4 \rightarrow s_7, s_4 \rightarrow s_8$) that will

be affected by the presence of the obstacle. These edges will be part of the CDG in the non-obstacle scenario but they should be removed if the obstacle is detected. Depending on the nature of the obstacle (i.e., communication or physical) the obstacle detection and the creation of the convex zone containing the obstacle will be performed as part of the CDG creation or later when the migration process is initiated.

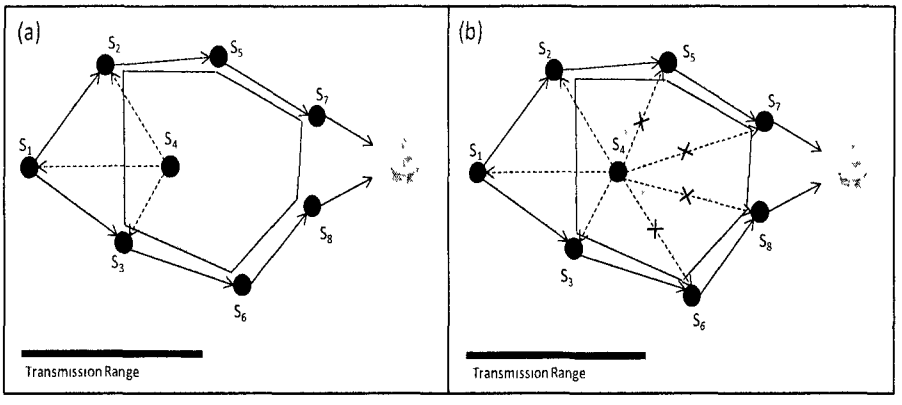


Figure 6.3: CDG construction with a single obstacle. (a) Communication obstacle, (b) physical obstacle.

If during the graph creation, a sensor encounters a communication obstacle (e.g., Figure 6.3(a)) sensor s_4 will not receive NEIGHBOUR_ACCEPT messages from any sensor. Moreover, sensor s_4 knows that the recharge station is not within range and there are no other sensors closer to the station (unless s_4 travels through the blackout area). Consequently, the only possible routes for s_4 would involved swapping positions with sensors s_1, s_2 or s_3. Sensor s_4 will reverse the orientation of these edges by sending a CHANGE_DIRECTION message to its former parents s_1, s_2, s_3. Only sensors that are completely trapped in a concave area (s_4) will send CHANGE_DIRECTION messages. Other sensors that received at least one NEIGHBOUR_ACCEPT message (e.g., s_2, s_3) will be completely oblivious of the presence of the obstacle.

Figure 6.3 (b) shows the construction of the CDG with a physical obstacle. Since

the sensors have limited visibility, the obstacle is not detected at CDG creation time The obstacle is only found when sensor s_4 attempts to swap positions with any of its graph neighbours (s_5, s_6, s_7, s_8) To avoid getting trapped in an unsuccessful swapping attempt, the sensors need to adapt their swapping process to what we call *cautious swapping* A cautious swapping is a swapping operation that can be canceled at any time if any of the party involves encounters an obstacle The swapping process will include an initial SWAP_REQUEST message followed by a SWAP_ACCCEPT message and a possible SWAP_CANCEL message if any of the sensors cannot complete the operation Upon receiving a SWAP_CANCEL message, the sensors will return to their original positions and delete the corresponding CDG edge For example, if s_4 finds the physical obstacle while travelling towards s_5, then it sends a SWAP_CANCEL and returns to its original position The sensors, then proceed to delete the edge $s_4 \rightarrow s_5$ from their neighbouring tables After cancelling all its swapping attempts, sensor s_4 realizes that it is trapped in a concave area and will then reverse the edges to its parents s_1, s_2, s_3 (i e , by sending CHANGE_DIRECTION messages) as the only possible routes around the physical obstacle

Further work in this area should evaluate the impact of the obstacle type and its detection mechanism (i e , early detection for communication obstacles or dynamic-late detection for physical obstacles) on the overall network survivability Furthermore, a sensor that falls into a panic run situation may no longer select the direct path towards the station (i e , sensor cannot travel through any type of obstacles) This implies that the shortest path around the convex area surrounding the obstacle should be selected A simple solution would be to greedily select the CDG neighbour with the smallest angle relative to the straight line to the recharge station However, more complex route selections are worth investigating

6.2.4 Open Problems

There are several unexplored variants of the facility absorbed FFP For example let $S = \{s_1, \quad , s_N\}$ a set of N sensors randomly deployed in an area of unspecified shape Let $P = \{p_1, \quad , p_K\}$ a set of points of interests and F, a static recharge facility In this instance of the FFP, all sensors s_i, $1 \leq i \leq N$, should visit a set of points $P_{s_i} \subset P$ and repeat this process continuously In other words, the sensors have to visit a sub-set of points of interest continuously and also visit the facility periodically to recharge their batteries

In this scenario, the following cases can be considered

1 Ordered vs non-ordered points of interests In this case the sensors have to visit a set of points in a given order or randomly but always guaranteeing the all points are visited before starting the next round of visits

2 Disjoint routes or itineraries $P_{s_i} \cap P_{s_j} = \emptyset, \forall s_i, s_j \in S$

3 Overlapping routes or itineraries $P_{s_i} \cap P_{s_j} \neq \emptyset, \forall s_i, s_j \in S$ This case should guarantee that two sensors do not visit the same point at the same time

4 Sensors can exchange their itineraries or point of interest At any point in time a pair of sensor can agree to exchange their itineraries

5 Sensors can "pick-up" other sensors' point of interest If a sensor dies of energy starvation, another sensor can add the depleted sensor's itinerary to its own

For all the aforementioned variants, the main goal is to achieve a state of equilibrium where the sensors fulfill their tasks but also cooperate to share a recharge station with limited resources (i e , number of recharge sockets)

Bibliography

[1] N. Aakvaag, M. Mathiesen, and G. Thonet, "Timing and power issues in wireless sensor networks - an industrial test case." *In Proceedings of the 2005 International Conference on Parallel Processing Workshops (ICPPW'05)*, pp. 419–426, 2005.

[2] M. I. Afzar, W. Mahmood, and A. H. Akbar, "A battery recharge model for wsns using free-space optics (fso)," *In Proceedings of the 12th IEEE International Multitopic Conference*, pp. 272–277, 2008.

[3] M. I. Afzar, W. Mahmood, S. M. Sajid, and S. Seoyong, "Optical wireless communication and recharging mechanism of wireless sensor network by using ccrs," *International Journal of Advance Science and Technology*, vol. 13, pp. 59–68, 2009.

[4] S. Ahn, Y. Lim, and J. Lee, "Adjusting the cluster size based on the distance from the sink." *In Proceedings of the First International Conference on High Performance Computing and Communications (HPCC 2005)*, vol. 3726, pp. 255–264, 2005.

[5] F. Arwin, K. Samsudin, and A. R. Ramli, "Swarm robots long term autonomy using moveable charger," *In Proceedings of the 2009 International Conference on Future Computer and Communication*, pp. 127–130, 2009.

[6] T. Z.-J. ation, W. Yi, and G. Zheng-Hu, "Eegfgr: An energy-efficient greedy-face geographic routing for wireless sensor," *In Proceedings of the 2007 IFIP international conference on Network and parallel computing*, pp. 171–182, 2007.

[7] R. Bachmann, F. J. Boria, P. G. Ifju, R Quinn, J. Kline, and R. Vaidyanathan, "Utility of a sensor platform capable of aerial and terrestrial locomotion," *In Proceedings of the 2005 IEEE/ASME International Conference on Advance Intelligent Mechatronics*, pp. 1581–1586, 2005.

[8] C. Bajaj, "The algebraic degree of geometric optimization problems." *Discrete Computational Geometry*, vol. 3, pp. 177–191, 1988.

[9] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia, "Routing with guaranteed delivery in ad hoc wireless networks," *Wireless Networks*, vol. 7, pp. 609–616, 2001.

[10] S. Capkun, M. Hamdi, and J. Hubaux, "Gps-free positioning in mobile ad-hoc networks," *In Proceedings of the 34th Annual Hawaii International Conference on System Sciences*, vol. 9, pp. 9008–9015, 2001.

[11] C. Y. Chang, C. T. Chang, Y. C. Chen, and S. C. Lee, "Active route-guiding protocols for resisting obstacles in wireless sensor networks," *IEEE Transactions on Vehicular Technology*, pp. 4425–4442, 2010.

[12] I. Chatzigiannakis, G. Mylonas, and S. Nikoletseas, "Modeling and evaluation of the effect of obstacles on the performance of wireless sensor networks," *In Proceedings of the 37th Annual Symposium on Simulation (ANSS'06)*, pp. 50–60, 2006.

[13] E. Cockayne and Z. Melzak, "Euclidean constructibility in graph-minimization problems." *Mathematical Magazine*, vol. 42, pp. 206–208, 1969

[14] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 2, pp. 243–255, 2004.

[15] A. Couture-Beil and R. Vaughan, "Adaptive mobile charging stations for multi-robot systems," *In Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1363–1368, 2009.

[16] S. Datta, C. Giannella, and H. Kargupta, "K-means clustering over a large dynamic network," *In Proceedings of 2006 SIAM Conference on Data Mining (SDM 06)*, pp. 153–164, 2006.

[17] S. Datta, I. Stojmenovic, and J. Wu, "Internal node and shortcut based routing with guaranteed delivery in wireless networks," *Cluster Computing*, vol. 5, pp. 169–178, 2002.

[18] A. Drenner and N. Papanikolopoulos, "Docking station relocation for maximizing longevity of distributed robotic teams," *In Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, pp. 2436–2441, 2006.

[19] W. Drytkiewicz, S. Sroka, and V. Handziski, "A mobility framework for omnet++." *In Proceesings of the 3rd International OMNeT++ Workshop*, 2003.

[20] C. Duan and H. Fan, "A distributed energy balance clustering protocol for heterogeneous wireless sensor networks," *In Proceedings of the 2007 International Conference on Wireless Communications, Networking and Mobile Computing*, pp. 2469–2473, 2007.

[21] L. Feeney, "An energy consumption model for performance analysis of routing protocols for mobile ad hoc networks," *Mobile Network Applications*, vol. 6, pp. 239–249, 2001.

[22] L. Feeney and M. Nilsson, "Investigating the energy consumption of a wireless network interface in an ad hoc networking environment," *In Proceedings of the*

20th Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Infocom 2001), vol. 3, pp 1548–1557, 2001

[23] P. A. Forero, A. Cano, and G. B Giannakis, "Consensus-based k-means algorithm for distributed learning using wireless sensor networks," *In Workshop on Sensors, Signal and Information Processing*, 2008.

[24] C. Frank and K. Romer, "Distributed facility location algorithms for flexible configuration of wireless sensor networks." *Proceedings of the 3rd IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS 2007)*, pp. 124–141, 2007.

[25] H. Frey, S. Ruhrup, and I. Stojmenovic, "Routing in wireless sensor networks," *Chapter4. Guide to Wireless Sensor Networks*, pp. 81–111, 2009.

[26] C. Garcia, C. Ibarguengoytia, J. Garcia, and J. Perez, "Wireless sensor networks and applications: a survey." *International Journal of Computer Science and Network Security*, vol. 7, pp. 264–273, 2007.

[27] E. Hamouda, N. Mitton, B. Pavkovic, and D. Simplot-Ryl, "Energy-aware georouting with guaranteed delivery in wireless sensor networks with obstacles," *Journal of Wireless Information Networks*, pp. 142–153, 2009.

[28] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Transactions on Wireless Communications*, vol. 1, no. 4, pp. 660–669, 2002.

[29] N. Heo and P K. Varshney, "Energy-efficient deployment of intelligent mobile sensor networks," *IEEE Transactions on Systems and Cybernetics*, vol. 35, no. 1, pp. 78–92, 2005.

[30] M. Hua, M. K. Lau, and J. Pei, "Continuous k-means monitoring with low reporting cost in sensor networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 12, pp. 1679–1691, 2009.

[31] R. Jain, A. Puri, and R. Sengupta, "Geographic routing using partial information for wireless ad hoc networks," *IEEE Personal Communication*, pp. 48–57, 2001.

[32] J. W. Jaromczyk and G. T. Toussaint, "Relative neighborhood graphs and their relatives," *In Proceedings of the IEEE*, vol. 80, pp. 1502–1517, 1992.

[33] X. Jianq, J. Polastre, and D. Culler, "Perpetual environmentally powered sensor networks." *In Proceedings of the Fourth International Symposium on Information Processing in Sensor Networks (IPSN 2005)*, vol. 7, pp. 463–468, 2005.

[34] M. Khalilian, N. Mustapha, N. Suliman, and A. Mamat, "A novel k-mean based clustering algorithm for high dimensional data sets," *In Proceedings of Multi-Conference of Engineers and Computer Scientists.*, pp. 488–492, 2010.

[35] K. Kouzoubov and D. Austin, "Autonomous recharging mobile robotics." *In Proceedings of 2002 Australian Conference on Robotics and Automation*, pp. 27–39, 2002.

[36] E. Kranakis, H. Singh, and J. Urrutia, "Compass routing on geometric networks," *In Proceedings of the 11th Canadian Conference on Computational Geometry*, pp. 51–54, 1999.

[37] D. Krivistki, A. Schuster, and R. Wolff, "A local facility location algorithm for sensor networks." *In Proceedings of the First IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS 2005)*, pp. 368–375, 2005.

[38] A. LaMarca, W. Brunnete, D. Koizumi, and M. Lease, "Making sensor networks practical with robots." *In Proceedings of the First International Conference Pervasive Computing (Pervasive-2002)*, vol. 2414, pp. 152–166, 2002.

[39] X. Li, A. Nayak, and I. Stojmenovic, "Exploiting actuator mobility for energy-efficient data collection in delay-tolerant wireless sensor networks." *In Proceedings of the Fifth International Conference on Networking and Services ICNS*, pp. 216–221, 2009.

[40] X. H. Li, K. L. Fang, L. Zhang, and J. He, "A clustering algorithm based on k-means for wireless indoor monitoring system," *In Proceedings of 2009 International Conference on Information Technology and Computer Science*, pp. 488–492, 2009.

[41] Y. Litus, R. Vaughan, and P. Zebrowski, "The frugal feeding problem: energy-efficient, multi-robot, multi-place rendezvous," *In Proceedings of the 2007 IEEE International Conference on Robotics and Automation*, pp. 27–32, 2007.

[42] Y. Litus, P. Zebrowski, and R. T. Vaughan, "A distributed heuristic for energy-efficient multirobot multiplace rendezvous," *IEEE Transactions on Robotics*, vol. 25, pp. 130–135, 2009.

[43] K. H. Low, G. Podnar, S. Stancliff, J. M. Dolan, and A. Elfes, "Operation robotic science boats using the telesupervised adaptive ocean sensor fleet system," *In Proceedings of IEEE International Conference on Robotics and Automation (ICRA 2008)*, pp. 1061–1068, 2008.

[44] C.-H. Lung and C. Zhou, "Using hierarchical agglomerative clustering in wireless sensor networks: An energy-efficient and flexible approach," *Ad Hoc Networks*, no. 3, pp. 328–344, 2010.

[45] C.-H. Lung, C. Zhou, and Y. Yang, "Applying hierarchical agglomerative clustering to wireless sensor networks," *In Proceedings of the International Workshop on Theoretical and Algorithmic Aspects of Sensor and Ad-hoc Networks (WTASA)*, pp. 97–105, 2007.

[46] J Luo and J. P. Hubaux, "Joint mobility and routing for lifetime elongation in wireless sensor networks," *In Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Infocom 2005)*, pp. 1735–1746, 2005.

[47] A. W. Matin and S. Hussain, "Intelligent hierarchical cluster-based routing," *In Proceedings of IEEE International Conference on Distributed Computing in Sensor Networks (DCOSS)*, pp. 165–172, 2006.

[48] Y. Mei, C. Xian, S. Das, Y. Hu, and Y. Lu, "Sensor replacement using mobile robots." *Computer Communications*, vol. 30, pp. 2615–2626, 2007.

[49] F. Michaud and E. Robichaud, "Sharing charging stations for long-term activity of autonomous robots," *In Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, pp. 2746–2751, 2002.

[50] N. Mitton, T. Razafindralambo, D. Simplot-Ryl, and I. Stojmenovic, "Hector is an energy efficient tree-based optimized routing protocol for wireless networks," *In Proccedings of the 4th International Conference on Mobile Ad-hoc and Sensor Networks (MSN'08)*, pp. 31–38, 2008.

[51] H. Mousavi, A. Nayyeri, N. Yazdani, and C. Lucas, "Energy conserving movement-assisted deployment of ad hoc sensor net- works," *IEEE Communications Letters*, vol. 10, no. 4, pp. 269–271, 2006.

[52] J. Mulder, X. Wang, F. Ferwerda, and M. Cao, "Mobile sensor networks for inspection tasks in harsh industrial environments," *Sensors*, vol. 10, pp. 1599–1618, 2010.

[53] R. Nelson and L. Kleinrock, "The spatial capacity of a slotted aloha multihop packet radio network with capture," *IEEE Transactions on Communications*, vol. 32, pp. 684–694, 1984.

[54] M. Norusis, "Cluster analysis," *Chapter 17. IBM SPSS Statistics 19 Guide to Data Analysis*, pp. 375–404, 2011.

[55] N. Patrikalakis, F. Hover, B. Ooi, H. Zheng, K. Yeo, W. Cho, T. Bandyopadhyay, A. Tan, H. Kurniawati, T. Taher, and R. R. Khan, "Infrastructure for mobile sensor network in the singapore coastal zone," *In Proceedings of the 20th International Offshore and Polar Engineering Conference*, 2010.

[56] W. Peng and D. J. Edwards, "K-means like minumum mean distance algorithms for wireless sensor networks," *In Proceedings of 2nd International Conference Computer Engineering and Technology (ICCET)*, pp. 120–124, 2010.

[57] M. Perillo, C. Zhao, and W. Heinzelman, "On the problem of unbalanced load distribution in wireless sensor networks," *In Proccedıngs of the Global Telecommunıcatıons Conference Workshops (GlobeCom 2004)*, pp. 74–79, 2004.

[58] M. Rahimi, H. Shah, G. Sukhatme, J. Heidemann, and D. Estrın, "Studying the feasibility of energy harvesting in a mobile sensor network." *In Proceedıngs of the IEEE Internatıonal Conference on Robotıcs and Automatıon*, pp. 19–24, 2003.

[59] S. Roundy, P. Otis, Y. Chee, J. Rabaey, and P. Wright, "A 1.9ghz rf transmit beacon using environmentally scavenged energy." *Dıgest IEEE Internatıonal Symposıum on Low Power Electrıcity and Devıces*, 2003.

[60] S. Ruhrup, H. Kalosha, A. Nayak, and I. Stojmenovic, "Message-efficient beaconless georouting with guaranteed delivery ın wireless sensor, ad hoc, and actuator networks," *IEEE/ACM Transactıons on Networkıng*, vol. 18, pp. 95–108, 2010.

[61] E. Saad, M. Awadalla, and R. Darwish, "Adaptive energy-aware gathering strategy for wireless sensor networks," *Internatıonal Journal of Computers*, no. 2, pp. 148–157, 2008.

[62] A. Schuster, D. Krivistki, and R. Wolff, "A local facility location algorithm for large-scale distributed systems." *Journal of Grıd Computıng*, vol. 5, pp. 361–378, 2007.

[63] M. Sharifi, S. Sedighian, and M. Kamali, "Recharging sensor nodes using implicit actor coordination in wireless sensor actor networks," *In Wıreless Sensor Network*, vol. 2, pp. 123–128, 2010.

[64] W. Sheng and G. Tewolde, "Robot workload distribution in active sensor networks." *In Proceedıngs of the Internatıonal Symposıum on Computatıonal Intelligence ın Robotıcs and Automatıon (CIRA 2007)*, pp. 224–229, 2007.

[65] V. Shnayder, B. Chen, and K. Lorincz, "Sensor network for medical care." *In Proceedıngs of the 3rd ınternatıonal conference on Embedded Networked Sensor Systems*, pp. 314–324, 2005.

[66] D. Simplot-Ryl, I. Stojmenovic, and J. Wu, "Energy efficient backbone construction, broadcasting, and area coverage in sensor networks." *Chapter 11. Handbook of Sensor Networks: Algorıthms and Archıtectures*, pp. 343–379, 2005.

[67] G. Song, Y. Zhou, F. Ding, and A. Song, "Mobile sensor networks system for monitoring of unfriendly environments," *Sensors*, pp. 7259–7274, 2008.

[68] I. Stojmenovic and X. Lin, "Power-aware localized routing in wireless networks," *IEEE Transactıons on Parallel and Dıstrıbuted Systems*, vol. 12, pp. 1122–1133, 2001.

[69] I Stojmenovic, M Russell, and B Vukojevic, "Depth first search and location based localized routing and qos routing in wireless networks," *In Proceedings of the 2000 International Conference on Parallel Processing*, pp 173–186, 2000

[70] G S Sukhatme, A Dhariwal, B Zhang, C Oberg, B Stauffer, and D A Caron, "The design and development of a wireless robotic networked aquatic microbial observing system," *In Environmental Engineering Science*, vol 24, no 2, pp 205–215, 2007

[71] K Supowit, "The relative neighborhood graph, with an application to minimum spanning trees," *Journal of the ACM*, vol 30, pp 428–448, 1983

[72] R Szewczyk, A Mainwaring, J Polastre, J Anderson, and D Culler, "An analysis of a large scale habitat monitoring application," *In Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, pp 214–226, 2004

[73] H Takagi and L Kleinrock, "Optimal transmission ranges for randomly distributed packet radio terminals," *IEEE Transactions on Communications*, vol 32, pp 246–257, 1984

[74] L Tan, Y Gong, and G Chen, "A balanced parallel clustering protocol for wireless sensor networks using k-means techniques," *In Proceedings of the 2nd International Conference on Sensor Technologies and Applications (SENSORCOMM 2008)*, pp 300–305, 2008

[75] G Tewolde and W Sheng, "Distributed multi-robot workload partition in manufacturing automation " *In Proceedings of the 4th IEEE Conference on Automation Science and Engineering*, pp 504–509, 2008

[76] T Tirta, B Lau, N Malhotra, S Bagchi, L Z , and Y Lu, "Controlled mobility for efficient data gathering in sensor networks with passively mobile robots " *Sensor Network Operations by Wiley-IEEE Press* , 2005

[77] A Vargas, "The omnet++ discrete event simulation system " *In Proceedings of the European Simulation Multi-conference (ESM'2001)*, pp 319–324, 2001

[78] G Wang, G Cao, and T L Porta, "Movement-assisted sensor deployment," *IEEE Transactions on Mobile Computing*, vol 5, no 6, pp 640–562, 2006

[79] W Wang, V Srinivasan, and K Vikram, "Extending the lifetime of wireless sensor networks through mobile relays," *IEEE/ACM Transactions on Networking*, vol 16, pp 1108–1120, 2008

[80] J Warwerla and R Vaughan, "Near-optimal mobile robot recharging with the rate-maximizing forager," *In Proceedings of the 9th European Conference on Artificial Life*, pp 776–785, 2007

[81] K. Xu, Y. Jia, and Y. Liu, "A novel hierarchical clustering routing algorithm for wireless sensor networks," *In Proceedings of the 2008 International Conference on Internet Computing in Science and Engineering*, pp. 282–285, 2008.

[82] S. Yang, M. Li, and J. Wu, "Scan-based movement-assisted sensor deployment methods in wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 8, pp. 1108–1121, 2007.

[83] O. Younis and S. Fahmy, "Heed: A hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks," *IEEE Transactions on Mobile Computing*, vol. 3, pp. 366–379, 2004.

[84] M. Z. Zamalloa, K. Seada, B. Krishnamachari, and A. Helmi, "Efficient geographic routing over lossy links in wireless sensor networks," *ACM Transactions on Sensor Networks*, pp. 1–33, 2008.

[85] G. Zhao, X. Liu, and A. Kumar, "Geographic multicast with k-means clustering for wireless sensor networks," *In Proceedings of IEEE Vehicular Technology Conference*, pp. 233 – 237, 2008.

[86] J. Zhao, A. Erdogan, and A. Tughrul, "A novel application specific protocol for wireless sensor networks." *In Proccedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 6, pp. 5894–5897, 2005.