EXPLOITING NODE MOBILITY FOR ENERGY OPTIMIZATION IN WIRELESS SENSOR NETWORKS

By

Fatme Mohammad El-Moukaddem

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Computer Science

2012

ABSTRACT

EXPLOITING NODE MOBILITY FOR ENERGY OPTIMIZATION
IN WIRELESS SENSOR NETWORKS

By

Fatme Mohammad El-Moukaddem

Wireless Sensor Networks (WSNs) have become increasingly available for data-intensive applications such as micro-climate monitoring, precision agriculture, and audio/video surveillance. A key challenge faced by data-intensive WSNs is to transmit the sheer amount of data generated within an application's lifetime to the base station despite the fact that sensor nodes have limited power supplies such as batteries or small solar panels. The availability of numerous low-cost robotic units (e.g. Robomote and Khepera) has made it possible to construct sensor networks consisting of mobile sensor nodes. It has been shown that the controlled mobility offered by mobile sensors can be exploited to improve the energy efficiency of a network.

In this thesis, we propose schemes that use mobile sensor nodes to reduce the energy consumption of data-intensive WSNs. Our approaches differ from previous work in two main aspects. First, our approaches do not require complex motion planning of mobile nodes, and hence can be implemented on a number of low-cost mobile sensor platforms. Second, we integrate the energy consumption due to both mobility and wireless communications into a holistic optimization framework.

We consider three problems arising from the limited energy in the sensor nodes. In the first problem, the network consists of mostly static nodes and contains only a few mobile nodes. In the second and third problems, we assume essentially that all nodes in the

WSN are mobile. We first study a new problem called *max-data mobile relay configuration (MMRC)* that finds the positions of a set of mobile sensors, referred to as *relays*, that maximize the total amount of data gathered by the network during its lifetime. We show that the MMRC problem is surprisingly complex even for a trivial network topology due to the joint consideration of the energy consumption of both wireless communication and mechanical locomotion. We present optimal MMRC algorithms and practical distributed implementations for several important network topologies and applications. Second, we consider the problem of minimizing the total energy consumption of a network. We design an iterative algorithm that improves a given configuration by relocating nodes to new positions. We show that this algorithm converges to the optimal configuration for the given transmission routes. Moreover, we propose an efficient distributed implementation that does not require explicit synchronization. Finally, we consider the problem of maximizing the lifetime of the network. We propose an approach that exploits the mobility of the nodes to balance the energy consumption throughout the network. We develop efficient algorithms for single and multiple round approaches. For all three problems, we evaluate the efficiency of our algorithms through simulations. Our simulation results based on realistic energy models obtained from existing mobile and static sensor platforms show that our approaches significantly improve the network's performance and outperform existing approaches.

ACKNOWLEDGMENTS

I would like express my gartitude to my advisors Dr. Eric Torng and Dr. Guoliang Xing for their constant guidance and useful critiques. I would also like to extend my thanks to Dr. Li Xiao, Dr. Abdol-Hossein Esfahanian, and Dr. Haydar Radha for serving on my committee. Finally, I wish to thank my family for their support and encouragement throughout my study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

Recent years have seen the deployment of WSNs in a variety of *data-intensive* applications including micro-climate and habitat monitoring [51], precision agriculture, and audio/video surveillance [37]. It is shown in [16] that a moderate-size WSN can gather up to 1 Gb/year from a biological habitat. Due to the limited storage capacity of sensor nodes, most data must be transmitted to the base station for archiving and analysis. However, sensor nodes must operate on limited power supplies such as batteries or small solar panels. Therefore, a key challenge faced by data-intensive WSNs is to minimize the energy consumption of sensor nodes such that the sheer amount of data generated within the lifetime of the application can be transmitted to the base station.

Several approaches have been introduced to reduce the energy consumption of nodes and prolong the lifetime of the network. In general, these approaches can be classified into four main groups: duty cycling, power control, data reduction and controlled mobility.

Duty cycling approaches reduce the energy consumption during a period of time by reducing the duty cycle of nodes, which is the fraction of time the node is turned on during

its lifetime. Nodes follow a sleep/wake up schedule. When in sleep mode, nodes enter in a low power consumption stage since one of the main energy consumption components (the radio) is turned off. When nodes wake up, they resume their data reception and transmission activities. These approaches take advantage of redundancy of sensor nodes in the deployment area; a single node can perform the same job of multiple nodes in a more energy efficient way. However, connectivity and coverage [4, 47, 29] issues arise when duty cycling is used. Nodes need to collaboratively set their sleep/wake up schedules to ensure data can successfully reach the sink. Different power management protocols have been proposed; they include on-demand [44, 66], scheduled rendez-vous [56, 34, 27] and asynchronous [55, 39, 68] strategies.

In power control approaches, the main idea is that nodes reduce their energy consumption by reducing their transmission power to the minimum levels needed. In [5, 25], the authors propose MAC protocols for power control in which nodes determine the minimum transmit levels needed based on several criteria such as channel gains and noise and interference levels at the receiver. In [32, 31, 6, 48], the authors propose different transmission power assignment schemes to reduce the total energy consumption while maintaining connectivity.

In data reduction approaches the goal is to reduce the amount of data that gets sent to the sink. This is usually done by reducing the amount of data that is generated by the sensors. In these cases, the energy consumed by the sensing component is reduced. In other cases, the strategy is to reduce the number of transmissions needed to relay data to the sink, and this reduces the energy consumed by the radio component. Different techniques have been proposed. The first is data aggregation [67, 30, 65]. This consists of in network processing of the data and generating a smaller size result to send to the sink as opposed to sending all the data gathered to the sink where processing takes place. This technique

is application specific; for example in an environment monitoring application, we might be interested in average temperatures so data aggregation can be used. The second technique is data compression. The data sensed is encoded into smaller size data at the sensors and then decoded at the sink [11, 60, 53]. The third technique is data prediction. This technique uses the data sensed to build a model describing data evolution. It can be done either at the sink or at the sources to predict (within some error bound) new data instead of sensing it. The last technique is efficient data acquisition which uses efficient sampling strategies. For example, adaptive sampling [10, 35] exploits temporal and/or spatial correlation of data to reduce the amount of data sensed. These approaches regularly measure the error margin between the actual data and the predicted data and dynamically adjust the frequency of actual data generation.

The last scheme for improving energy efficiency is using mobility. Recent work showed that the energy cost of WSNs can be significantly reduced by utilizing the mobility of nodes. Several different approaches have been proposed. A robotic unit may move around the network and collect data from static nodes through one-hop or multi-hop transmissions [15, 36, 63, 26, 64]. The mobile node may serve as the base station or a "data mule" that transports data between static nodes and the base station [24, 46, 23]. Mobile nodes may also be used as *relays* [62] that forward the data from source nodes to the base station. Several movement strategies for mobile relays have been studied in [62, 17].

Although the effectiveness of mobility in energy conservation is demonstrated by previous studies, the following two key issues have not been addressed by any mobility technique. First, the movement cost of mobile nodes is not accounted for in the total network energy consumption. Instead, mobile nodes are often assumed to have replenishable energy supplies.

For instance, a mobile node may periodically recharge its battery at a fixed charging dock [26]. However, energy replenishment is not always feasible due to the constraints of the physical environment. Second, complex motion planning of mobile nodes is often assumed in existing solutions which introduces significant design complexity and manufacturing costs. In [50, 18, 26, 64], mobile nodes need to compute optimal motion paths and continuously change their orientation and/or speed of movement. Such capabilities are usually not supported by existing low-cost mobile sensor platforms. For instance, Robomote [12] nodes are designed using 8-bit CPUs and small batteries that only last for about 25 minutes in full motion. Complex motion planning is not practical for such platforms due to the limited computational power and short battery lifetime.

We explore the use of low-cost mobile sensor nodes to improve the energy efficiency of a WSN. Different from mobile base station or data mules, mobile nodes do not transport data; instead, they move to different locations and then remain stationary to forward data along the paths from sources to the base station. As a result, the communication delays can be significantly reduced compared with using mobile stations or data mules. We propose different approaches to reduce the total energy consumption of data-intensive WSNs, to maximize the efficiency of the network in gathering data and to maximize the network lifetime. Our approaches are motivated by the current state of mobile sensor platform technology. On the one hand, numerous low-cost mobile sensor prototypes such as Robomote [12], Khepera [1], and FIRA [28] are now available. Their manufacturing cost is comparable to that of typical static sensor platforms. As a result, they can be massively deployed in a network. Our approaches take advantage of this capability by assuming that we have a large number of mobile sensor nodes. On the other hand, due to low manufacturing

cost, existing mobile sensor platforms are typically powered by batteries and only capable of limited mobility. Consistent with this constraint, our approaches only require simple motions of mobile relays, i.e., one-shot relocation to designated positions after deployment. Compared with our approach, existing mobility approaches (such as mobile base station and data mule) typically assume a small number of powerful mobile nodes.

We note that all our formulations are based on realistic energy consumption models of both node movement and wireless communications that we developed using empirical results obtained from two widely used radios (CC2420 [3], CC1000 [2]) which is in contrast to existing mobility approaches that only account for the communication energy consumption. We make the following contributions.

1. We leverage mobility of sensor nodes to improve the performance of wireless sensor networks. We consider different criteria for measuring the performance of the network such as the data gathering capacity, the total energy consumption and the network's lifetime.

2. For each performance criterion, we present a new problem formulation. We develop optimal algorithms as well as fast heuristics to reconfigure the positions of the mobile nodes in the network such that the performance is greatly improved. We also propose practical distributed implementations for all our algorithms. We show that these algorithms have a low overhead as they converge to the final configuration after only a few iterations and they require little synchronization between nodes.

3. We conduct extensive simulations based on the realistic energy models we adopted. We show that our algorithms outperform existing non mobility solutions and significantly

improve the network's energy efficiency. Moreover, our distributed implementations are shown to have fast convergence and low messaging overhead.

The rest of this thesis is organized as follows. Chapter 2 reviews existing mobility approaches. In Chapter 3, we introduce the network model and develop models for energy consumption of different operations such as communication and mobility. In Chapter 4, we consider networks consisting of mostly static nodes and containing a small number of mobile nodes. We define the problem of maximizing the efficiency of a network by maximizing the total amount of data that the sink can gather throughout the lifetime of the network. We propose optimal algorithms as well as heuristics that produce close to optimal results. In Chapter 5, we define the problem of minimizing the total energy consumption in networks consisting entirely of mobile nodes and we propose iterative algorithms for the single-flow and multiple flows problems. We also show the advantage of our approach through extensive simulation results. In Chapter 6, we consider the problem of maximizing the network lifetime. We propose algorithms to balance the energy consumption across the network and show through simulations that the lifetime is substantially increased. We conclude the thesis in Chapter 7.

CHAPTER 2

Background

Several approaches have been used to reduce the energy consumption in battery operated wireless sensor networks. They include duty cycling strategies, data driven strategies and mobility. In this chapter, we review mobility strategies for reducing energy consumption. These strategies can be divided further into three classes: mobile base stations, data mules and mobile relays. In the following sections, we discuss each of these classes.

2.1 Mobile Base Stations

One of the main energy efficiency challenges of static wireless sensor networks is that nodes closer to the base station deplete their energy at a faster rate than other nodes since they have to transmit more data. One remedy is to use a mobile base station. When the base station relocates to a new position, it reduces the traffic load of the nodes close to its old position and increases the load at nodes close to its new position. In general, mobile base station approaches aim at balancing the energy consumption in the network by constantly rotating which nodes are close to the base station.

In some work, all the nodes are always performing multiple hop transmissions to the base station regardless of its position [15, 36, 63]. In [15], the authors divide the (known) lifetime of the network into equal periods of time called rounds. Base stations are relocated at the beginning of every round using a linear program with the objective of either minimizing the total energy spent at the nodes or minimizing the energy spent at any node in the network. In [36], the authors claim that the best mobility strategy for the base station is to move along the periphery of the network (assuming nodes are placed in a circular area) because this strategy minimizes the average load on any node. The authors also propose a different mobility strategy that takes into consideration the routing paths. This heuristic shortens the trajectory of the base station to a smaller circle inside the network area and combines straight and round routing paths from the sensors towards the base station. In [63], the authors also use a linear programming formulation for finding the mobility paths of the sink. Unlike [15, 36], their objective is to maximize the system lifetime defined as the time until the first node dies because of energy depletion. In both [15, 63], the base station can only move to a predefined and finite set of locations.

In other work, nodes only transmit to the base station when it is close to them (or a neighbor). The goal is to compute a mobility path to collect data before any node experiences a buffer overflow [50, 18, 26]. These approaches incur high latencies due to the low to moderate speed, e.g. 0.1-1 m/s [12, 50], of mobile base stations. In [50], each node in the network has an overflow period based on its data generation rate and buffer that is known in advance. The traveling time between any two nodes is also known. The authors prove that the problem of scheduling visits to static nodes such that no node's buffer overflows is NP-Complete. They propose several heuristics such as earliest deadline first, earliest deadline

first with k-lookaheads, greedy based on the weighted sum of two criteria (earliest deadline and closeness to the sink) as well as known heuristics for solving vehicle routing problems with time windows. In [18], the authors present an algorithm for finding a route for the mobile base station that visits all the nodes in the network, some more frequently than others, such that no node's buffer overflows between visits. The main idea of the algorithm is to divide nodes into bins according to their geographic location and their overflow time. In each bin, a route that visits each node once is computed using a solution for the Traveling Salesperson Problem. These routes are then concatenated to form a schedule for visiting all the nodes in the network. Once this schedule is found, the speed of the base station is adjusted to make the route feasible. In [26], the authors propose a solution that achieves a balance between the latency of relaying data to the base station and the total energy consumption at static nodes. Their approach consists of dividing static nodes into clusters. In each cluster, data is transmitted to the head node. The mobile base station then follows a shorter path that only communicates with the cluster heads.

2.2 Data Mules

Data mules are similar to mobile base stations [24, 46, 23] in that the mobile node mechanically carries the data. Data mules pick up data from the sensors and transport it to the sink. Similar to mobile base stations, data mules introduce large delays. First, sensors have to wait for a mule to pass by before starting their transmission. Second, data mules take a long time to then deliver the data to the sink due to their limited speed.

In [38], the data mule visits all the sources to collect data, transports data over some distance, and then transmits it to the static base station through the network. The goal is to

find a movement path that minimizes both communication and mobility energy consumption. The data mule has a sequence of tasks to perform. The plane is divided into regions such that each request falls in a different region. Each region is divided using a two dimensional grid resulting in potential points the mule can visit. The data mule then picks a point from each region to perform its corresponding task using Dijkstra's shortest path algorithm.

Similar to [26], the approach in [64] achieves a balance between energy savings and data collection delay. Additionally, it takes into consideration the deadline requirements for relaying data to the base station. The goal is to find a set of static nodes in the network at which the mobile nodes meet to pick up the data and transport it to the base station. The authors propose an optimal solution when the mobile node path is constrained to the routing tree and a utility based greedy algorithm when the path is unconstrained.

2.3 Mobile Relays

In the third approach, the network consists of mobile relay nodes along with static base station and data sources. Relay nodes do not transport data; instead, they move to different locations to decrease the communication costs. Goldenberg et al. [17] showed that an iterative mobility algorithm where each relay node moves to the midpoint of its neighbors converges on the optimal solution for a single routing path. However, they do not take into consideration the cost of moving the relay nodes. In [52], the authors observe that the optimal positions for nodes participating in a single flow transmission are the evenly spaced positions along a straight line between the source and the sink. They propose a mobility algorithm in which each node iteratively moves to midpoint of its neighbors. In this approach, mobile nodes decide to move only when moving is beneficial, (i.e., mobility costs

are covered by the savings in transmission costs). It is shown that this strategy converges to the optimal positions. However, the only target point considered is the midpoint of the neighbors whereas other points may be better when we consider the energy consumed by movement of the node.

2.4 Discussion

All three problems that we consider aim at efficiently using the limited energy available in the network by exploiting the mobility of nodes. They all fall in the class of mobile relays. However, they differ in the network setting, the specific objective they each optimize, and subsequently in the approach we propose to reach that objective. In the first problem, MMRC, the network is mostly static and contains only a small number of mobile nodes. A fraction of the static nodes generates data; the rest are used as relays. We study how to exploit this small number of mobile nodes to maximize the total amount of energy gathered during the lifetime of the network. We propose inserting the mobile nodes into the routing lines to help the weak links in the networks. The second problem we propose considers a similar setup as MMRC in that only a fraction of the nodes generates data but differs in that it considers networks that entirely consist of mobile nodes. The objective is to minimize the total amount of energy consumed by all nodes in the network when relaying data from the source nodes to the sink. We propose an approach that first constructs a routing tree that includes both nearby and distant nodes. Then, our approach collectively moves all the mobile relay nodes in the tree closer together while maintaining the connectivity of the network. The third problem, maximizing the network lifetime, also considers networks entirely composed of mobile nodes. Additionally, it considers scenarios in which all the nodes generate data

that need to be relayed to the sink. We propose to rotate the nodes within the network one or multiple times to balance the energy consumption across different nodes.

A general advantage of our approaches is that we keep a relatively stable routing structure; namely, we reduce the disruptions to the network topology caused by mobility. In MMRC, only a few nodes are inserted into the routing topology and each insertion affects exactly two neighbors. In our solution to the energy minimization problem, once the routing tree is constructed, its exact structure is maintained throughout the moving process. In our final approach, node rotation, the routing topology is maintained with respect to the locations of nodes.

Compared to the approaches of mobile base stations and data mules, all our approaches consider the energy consumption of both mobility and transmission. Additionally, with the exception of the multiple rotation solution to the network lifetime problem, they all require much simpler motion planning. Specifically, after deployment, each mobile relay relocates only once and then remains stationary for data forwarding. Such a simple mobility strategy can be implemented on a number of existing mobile sensor platforms [12, 1, 50].

Our proposed solution to the total energy minimization problem is similar to the mobile relay schemes in [17] and [52]. But one key difference between our approach and those schemes is that we consider all possible locations as possible target locations for a mobile node instead of just its current position and the midpoint of its neighbors. Our approaches (specifically for reducing the total energy consumption and maximizing data gathering capacity) accurately compute the optimal target position of each mobile relay node. Moreover, we allow nodes to behave differently. If it is beneficial for one node to move, then this node moves independently of other nodes. We observe that any local improvement is a global

improvement. In each transmission round, different subsets of nodes might move to new locations while the rest remain in place. Our approaches, in this aspect, are flexible to accommodate for additional mobility constraints on individual nodes.

CHAPTER 3

System Model

In this chapter, we introduce the model abstracting the network behavior and properties. We also define the energy consumption models of several components of sensor nodes.

3.1 Network Model

We consider wireless sensor networks in which nodes are randomly deployed in a two dimensional field. We consider all links to be bidirectional. Furthermore, we assume that all nodes know their locations either by GPS units mounted on them or by a localization service in the network so any node can determine the euclidean distance separating it from any other node. We model the network as an undirected graph G in which vertices represent network nodes. Each vertex v is placed in the 2-dimensional Euclidean plane. An edge uv exists in G if u and v are within communication range from each other so it is possible to establish a communication channel between both nodes. These communication channels are determined as a connected subset of the existing edges by the routing algorithm that we adopt in each application. We assume that all communication channels have low interference and low data

loss rate as the communication power level is set such that the signal reception is strong and reliable as explained in Section 3.2.2.

3.2 Energy Consumption Models

Sensor nodes consume energy during various operations such as communication, computation, movement and sensing. However, communication and mobility energy consumption are the two most critical sources of energy consumption and battery depletion. In this work, we develop models for the energy consumed by nodes during transmissions and receptions, and we adopt energy models for wheeled sensor nodes to model energy consumed by nodes during movement, all of which facilitate our holistic objective in all the problems that we consider. This advances significantly upon previous work that considers only the energy consumed during transmissions. We do not consider the energy consumed in idle listening states as it can be significantly reduced through existing sleep scheduling schemes [59]. Alternatively, we can view our results as the limits of what can be achieved assuming a perfect sleep scheduling algorithm.

3.2.1 Mobility

First, we define the mobility energy consumption models. We consider wheeled sensor nodes with differential drives such as Khepera [1], Robomote [12] and FIRA [28]. This type of node usually has two wheels, each controlled by an independent engine. The direction and speed of the node are determined by the angular velocity and direction of each wheel. We adopt the distance proportional energy consumption model which is appropriate for this

kind of node [58]. The energy $E_M(d)$ consumed by moving a distance d is modeled as:

$$E_M(d) = kd$$

The value of the parameter k depends on the speed of the node. In general, there is an optimal speed at which k is lowest. The parameter k increases as the difference between the node velocity and the optimal velocity increases. In [58], the authors discuss in details the variation of the energy consumption with respect to the speed of the mote. When the node is running at optimal speed, $k = 2\ J/m$ [58]. In our simulations, we use $k = 1, 2, 3$ and 4 to model different speeds and different terrains.

3.2.2 Communication

We develop our transmission energy consumption model by analyzing empirical results obtained from two radios CC2420 [3] and CC1000 [2] that are widely used on sensor network platforms. For CC2420, the authors of [45] studied the power needed for transmitting packets reliably (e.g., above 95% packet reception ratio) over different distances. Let $E_T(d, m)$ be the energy consumed to transmit m bits reliably over distance d and $E_R(m)$ the energy consumed by receiving m bits. We have

$$E_T(d, m) = m(a' + bd^2)$$

$$E_R(m) = ma''$$

where a', a'' and b are constants depending on the environment. We assume that a node only receives data when it has enough energy to transmit that data to the next node. In this case,

a node receiving m bits and transmitting it over a distance d consumes $m(a' + a'' + bd^2) = m(a + bd^2)J$.

We now discuss the instantiation of the above model for the CC2420 and CC1000 radio platforms. We used measurements in [45] on CC2420 obtained in an outdoor environment under the default parameters to determine the power level needed for different distances to get a packet reception ratio higher than 95%. The same measurements were collected for the CC1000 platform. Using these values along with current and voltage requirements from [3] and [2], we computed the transmission energy consumption model parameters as summarized in table 3.1. These values are consistent with the theoretical analysis in [20]. We observe that $a'' > a'$. On the other hand, there is no dependence on distance d for reception whereas there is a dependence on d for transmission.

To capture a wider range of motes and environment conditions, we varied the values for parameters a', a'', and b in our simulations over an interval containing the values we empirically obtained (Table 3.1).

Table 3.1: Parameter values for energy consumption during communication

	a' J/bit	a" J/bit	b Jm^{-2}/bit
CC2420	0.6×10^{-7}	1.4×10^{-7}	4.0×10^{-10}
CC1000	0.3×10^{-7}	2.6×10^{-7}	2.0×10^{-10}

We also note that although the mobility parameter k is roughly 10^{10} times larger than the transmission parameter b, the relay node does not move much whereas large amounts of data are transmitted. For large enough data chunk sizes, the savings in energy transmission costs compensate for the energy expended to move the nodes resulting in a decrease in total energy consumed.

CHAPTER 4

Maximization of Data Gathering Capacity Using Mobile Nodes

In this chapter, we propose to use low-cost disposable mobile nodes to increase the ability of a network to collect data and transmit it to the sink. We consider the following scenario. We assume that we have a network consisting of static nodes that generate or relay data to the sink. We also assume that we have a small number of mobile nodes that were added to the network. We propose an approach that uses these mobile nodes to alleviate bottleneck transmission links and thus increase the network's data gathering capacity.

We first present the formulation of a new problem called *Max-Data Mobile Relay Configuration (MMRC)* for hybrid WSNs composed of both static and mobile nodes. The objective of MMRC is to configure the positions of mobile relays to maximize the data gathering capacity of WSNs, which is defined as the total amount of data that can be transmitted to the BS during the lifetime of a network topology. We define k-MMRC as the variant in which at most k mobile relay nodes may join one link to improve its data gathering capacity. Our

formulation is based on realistic energy consumption models of both wireless communications and node movement, which is in contrast to existing mobility approaches that only account for the communication energy consumption. We then develop an optimal algorithm for the base case of the 1-MMRC problem with one mobile and two static nodes. Despite the trivial network configuration, we show that the base case of 1-MMRC has a surprisingly high complexity. In particular, a direct analytical solution requires finding the roots of a degree six bivariate polynomial. To the best of our knowledge, this result is the first to reveal the complexity of data gathering capacity maximization when both communication and mobility energy consumption are jointly considered. Based on the algorithm to the base case of 1-MMRC, we develop optimal algorithms for several important network topologies including lines, stars, and trees with and without data reduction techniques. We also present a practical distributed implementation for these algorithms. Moreover, we consider the 2-MMRC variant of the problem and develop efficient algorithms for the base case and general topologies. Finally, we conduct extensive simulations and show that our algorithms can increase the amount of data gathered during the system lifetime by a factor of 2 or more. We also show that our algorithms for 1-MMRC achieve similar data gathering capacities as our 2-MMRC algorithms.

The rest of this chapter is organized as follows. In Section 4.1, we formally define the MMRC problem. Section 4.2 presents several optimal algorithms for 1-MMRC along with a practical distributed implementations. In Section 4.3, we propose a solution for the 2-MMRC variant. Section 4.4 describes our simulation results and Section 4.5 summarizes the chapter.

4.1 Problem Definition

The fundamental problem we address in this chapter is maximizing the amount of data that can be gathered by the base station of hybrid WSNs composed of mostly static nodes and a small number of mobile nodes. A key challenge of utilizing low cost mobile relay nodes is that they often are more unreliable than static nodes. For example, mobile relay nodes often experience mechanical failures when moving in rough terrain. To provide a balance between maximizing data gathering capacity and mitigating the impact of mobile sensor nodes on network reliability, we make the following assumptions. We first assume that a routing topology connecting the static nodes to the sink has already been established. We also assume that each mobile relay can help at most one link. These assumptions allow mobile relays to improve the system's data gathering capacity but minimize the dependence of the communication on mobile nodes.

We define the data gathering capacity of a network to be the total amount of data that can be transmitted from the sources to the sink during the lifetime of the current routing topology. Consistent with several other papers [49, 9, 43] the lifetime of the current topology ends when the first node dies. We use this definition because the death of any node entails a reconfiguration of the underlying routing topology of static nodes and a consequent reconfiguration of the mobile nodes. However, we do assume that data that is in transit from a source to the sink and still has a viable path to the sink will be sent along the remaining viable links to the sink. For example, suppose a node u expends all its energy to transmit data to an intermediate node v. Since u has expended all its energy, the system lifetime has ended. However, we assume that node v and other nodes on the path to the

sink can still send this data to the sink assuming they have enough energy. We use this data gathering capacity metric as an alternative to two other important but conflicting metrics, lifetime and throughput, since it directly measures the network's ability to perform its target functionality of transmitting data from the sources to the sink. In one variant with a star routing topology where each source node is directly connected to the sink node, we simply measure the raw data gathering capacity without regard to system lifetime as no reconfiguration will be performed. That is, we simply measure how much data can be transmitted from the sources to the sink until the single sink node dies or all the source nodes die.

We define the k-MMRC problem as follows. Given a network N consisting of a large number of static nodes and a small number of mobile nodes, and the routing topology connecting the static nodes to the sink, the goal is to find the optimal positions of the mobile nodes to join the transmission routes such that the total data gathering capacity of N is maximized and at most k relay nodes join each transmission link. In this definition, we limit the number of mobile relays that can help a single link to the parameter k for two main reasons. First, since the number of mobile nodes in the network is relatively small, we want to allow a fair number of links to be helped. Second, as mobile nodes may be unreliable, we want to reduce the dependence of a single link on a large number of mobile nodes. We consider two variants of the problem, with $k = 1$ and 2. Our results show that it is enough to solve the 1-MMRC variant as it results in comparable improvements in data gathering capacity to the 2-MMRC variant.

Different from previous work that only accounts for the energy required for transmission, we take into account the energy required to transmit and receive messages plus the energy

required to move the mobile relay nodes into position. We use the energy models defined in chapter 3. These model parameters fit the quadratic path loss model for distances up to 35m. For larger transmission distances, the energy consumption increased and diverged from the quadratic model and could be modeled by higher path loss coefficients (4 and 6) as discussed in [61]. To cope with higher path loss coefficients, the only modification we need is a new optimal point location algorithm to replace our algorithm in Section 4.2.1.

4.1.1 Max-Data Mobile Relay Configuration

We consider four variants of our problem that cover three different topologies and several different metrics.

(1) MMRC in a line: The static nodes form a line between the source and the sink. We want to maximize the total amount of data that can be relayed from the source to the sink before any node runs out of energy. This is equivalent to maximizing the bottleneck capacity of any transmission link on the line.

(2) MMRC in a star: The star transmission topology consists of n static sources and one static sink. Each of the n source nodes transmits data directly to the sink. We again want to maximize the total amount of data that can be sent from all n sources to the sink. The star topology is widely used in networks running clustering protocols and is the primary topology supported by the 802.15.4 standard [22]. For the star topology, we measure the total amount of data gathered until either the sink node dies or all the source nodes die since we will not reconfigure the topology when any source node dies.

(3) Fair MMRC in a tree with data reduction: The static nodes form a tree with n source nodes sending data through the tree to a single sink. When an event occurs, each source

node sends one data unit to its parent. To ensure fairness, we require that each source's data must reach the sink. We assume perfect data aggregation where any internal node is able reduce the amount of data it sends to its parent to one data unit per event. Many natural sensor network applications use such an N-to-1 aggregation model. For instance, a user may periodically query the maximum or average value of temperature readings from all sensors deployed in a large biological habitat. Moreover, many in-network information processing techniques such as data fusion [57] can aggregate noisy measurements or local decisions from multiple sensors to improve the accuracy of sensing. In both applications described above, N sensor readings can be aggregated into one, which significantly reduces the energy consumption of the network. The net result is that we must maximize the bottleneck capacity of any link in the tree.

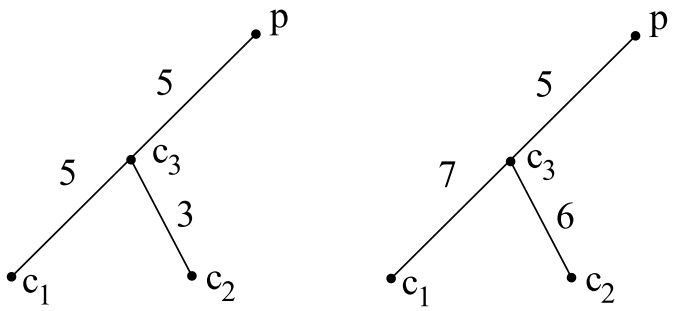

Figure 4.1: Two static tree topologies; each link is labeled with its capacity.

Consider the two examples in Figure 4.1. In the left tree, the data for 3 events can be sent to the sink before source c_2 runs out of energy; in the right tree, the data for 5 events can be sent to the sink before node c_3 runs out of energy. In both cases, the number of events reachable to the sink is limited by the edge with the smallest capacity.

(4) Fair MMRC in a tree without data reduction: this problem is identical to the previous problem except no data reduction is possible. That is, if an internal node receives one data

unit from 3 children, it must send 3 data units to its parent. In both trees in Figure 4.1, after 2 events are processed, c_3 has only enough energy to send one data unit to the sink. With our fairness constraint, no more events can be processed as a third event would require sending 2 data units to the sink.

4.2 1-MMRC: Single Relay per Link

The MMRC problem of maximizing system data gathering capacity is much more complex than the seemingly similar problem of minimizing the total energy required to transmit a fixed amount of data in a network [14] because we must account for the energies consumed by each individual node relative to its initial energy rather than worrying only about the total energy consumed by all nodes together. To the best of our knowledge, we are the first to optimize data gathering capacity taking into account the energy consumed by both movement and communication. In this section, we consider the special case in which a link may be helped by at most one relay. We first present an optimal algorithm for the simplest possible "base case" of the 1-MMRC problem: a topology with only one source, one sink, and one mobile relay. Next, we discuss the assignment problem, a weighted matching problem that several of our application problems can be reduced to. We then present optimal algorithms for our four variants of 1-MMRC.

4.2.1 Base Case of 1-MMRC

Our base case consists of computing the optimal position r^* to move relay node r to that will maximize the total number of bits that can be sent from static source node s_1 to static sink node s_2 using r as a relay at position r^* given initial energies e_1, e_2, and e_r.

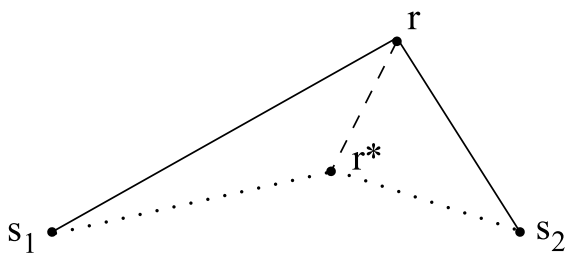

Figure 4.2: Example scenario illustrating the base case. Using relay r at position r^* increases the amount of data that can be sent.

An example scenario is given in Figure 4.2. For cases where it may be better to not use r because r has low energy, we assume that s_1 sends data directly to s_2. Note that we focus on maximizing the amount of data gathered by s_2 during the lifetime of the current network topology. Therefore, we do not allow s_1 to send data to r until r dies and then have s_1 send data directly to s_2 because this requires a topology reconfiguration. In the rest of this section, we study the problem of finding r^* when it is helpful to use r.

Optimal Algorithm

We now show that a direct analytical solution to the base case requires finding the roots of a degree 6 bivariate multi-modal polynomial. For any candidate point r', we define two key values: $m(s_1, r')$ which is the amount that s_1 can send to r located at r', and $m(r', s_2)$ which is the amount that r can send to s_2 when r is located at r' taking into account the energy r loses by moving to r'. It follows that the maximum amount of data s_1 can send to s_2 using r as a relay at r', denoted $m(r')$, is the minimum of $m(s_1, r')$ and $m(r', s_2)$. Optimal positions r^* have one of the following properties:

(O1) $m(s_1, r^*) = m(r^*, s_2)$ or

(O2) $\forall r', m(s_1, r^*) \geq m(s_1, r')$ and $m(s_1, r') < m(r', s_2)$ or

(O3) $\forall r', m(r^*, s_2) \geq m(r', s_2)$ and $m(s_1, r') > m(r', s_2)$

Let's assume condition (O1) holds where r^* is located at (x, y) and (x_1, y_1), (x_r, y_r), and (x_2, y_2) are the initial coordinates of nodes s_1, r and s_2, respectively. Let $d(u, v)$ be the distance between two nodes u and v. Finding point r^* is equivalent to solving the following equation.

$$m(s_1, r^*) = m(r^*, s_2) \tag{4.1}$$

$$\Leftrightarrow \frac{e_1}{a_1 + b_1 d(s_1, r^*)^2} = \frac{e_r - k_r d(r, r^*)}{a_r + b_r d(r^*, s_2)^2} \tag{4.2}$$

$$\Leftrightarrow A(x, y)B(x, y) + C(x, y) = 0 \tag{4.3}$$

$$\Rightarrow B(x, y)^2 A(x, y)^2 - C(x, y)^2 = 0 \tag{4.4}$$

where

$$\begin{aligned} A(x, y) =& k_r \left(a_1 + b_1(x - x_1)^2 + b_1(y - y1)^2\right) \\ B(x, y) =& \sqrt{(x - x_r)^2 + (y - y_r)^2} \\ C(x, y) =& e_1 \left(a_r + b_r(x - x_2)^2 + b_r(y - y_2)^2\right) \\ & - e_r \left(a_1 + b_1(x - x_1)^2 + b_1(y - y_1)^2\right). \end{aligned}$$

Because $A(x, y)$ and $B(x, y)^2$ have degree 2, equation (4.4) has degree 6; this equation is also a nonconvex (multi-modal) polynomial. Thus, solving the base case using analytical methods is numerically complex.

We present an efficient algorithm that converges on an optimal solution by solving the special case of the problem in which we are given the direction that r will move. Let α be any point on the line segment between s_1 and s_2 including s_1 and s_2, let $pos_r(\alpha)$ be the optimal position for r when it moves towards α, and let $\overline{M}(\alpha)$ be the resulting amount of data that can be transmitted from s_1 to s_2 when r moves to $pos_r(\alpha)$. We develop an algorithm that computes $pos_r(\alpha)$ and $\overline{M}(\alpha)$ for any α between s_1 and s_2, and we prove

that $\overline{M}(\alpha)$ is unimodal on this range. Thus, we can converge on the global optimum by performing a golden ratio search for α between s_1 and s_2.

We now discuss the algorithm for computing $pos_r(\alpha)$ and $\overline{M}(\alpha)$. We first consider optimality conditions (O2) and (O3) by finding the unique points r_1 and r_2 on the line to α that maximize $m(s_1, r')$ and $m(r', s_2)$, respectively. Point r_1 is returned if $m(r_1, s_2) > m(s_1, r_1)$ as condition (O2) applies. Otherwise, if $m(r_2, s_2) < m(s_1, r_2)$, then we return r_2 as condition (O3) applies. Otherwise, it must be the case that condition (O1) applies. When restricting the movement to a given line (with slope s_α), the distance traveled becomes $\sqrt{1+s_\alpha^2}|x - x_r|$ and equation (4.2) becomes

$$\frac{e_1}{a_1 + b_1(x - x_1)^2 + b_1(y - y1)^2} = \frac{e_r - k_r\sqrt{1+s_\alpha^2}|x - x_r|}{a_r + b_r(x - x_2)^2 + b_r(y - y2)^2}$$

Identifying such points is equivalent to solving a degree 3 polynomial. Thus, there are at most 3 possible points, all of which can be computed in $O(1)$ time. We return the point that maximizes the data sent. In summary, we find the optimal position along a given line by examining at most five well-defined points.

This base case algorithm only converges on the optimal r^* for energy models with path loss coefficient 2 or smaller. For other energy models with higher path loss coefficients, we need to use different techniques to find the optimal relocation point. We do not have efficient algorithms that do this for arbitrary energy models. We now prove that $\overline{M}(\alpha)$ is unimodal.

Theorem 1. *$\overline{M}(\alpha)$ is unimodal for $\alpha \in \overline{s_1 s_2}$.*

Proof. We define the function $g_r(\alpha)$ to be the value of the maximum message that can be

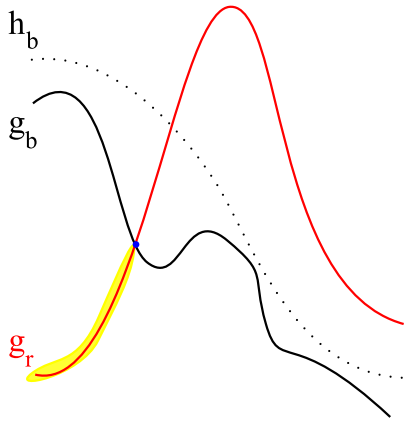

Figure 4.3: The highlighted portion of $g_r(\alpha)$ is part of $\overline{M}(\alpha)$

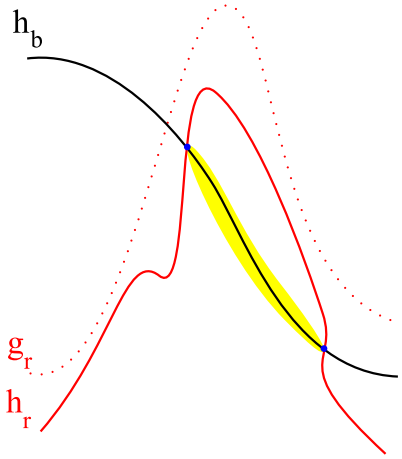

Figure 4.4: The highlighted portion of $h_b(\alpha)$ is part of $\overline{M}(\alpha)$

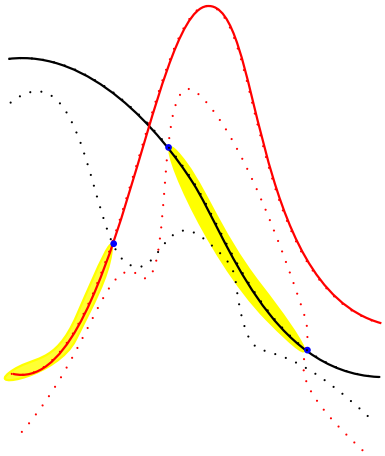

Figure 4.5: The set of point $R \cup B$ form a unimodal set

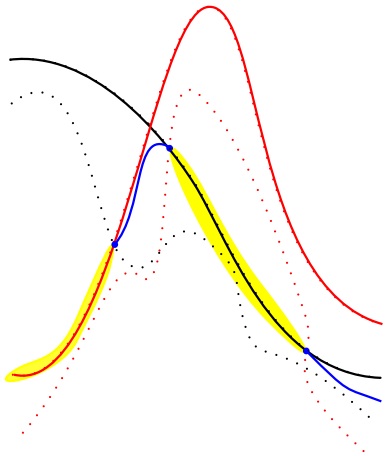

Figure 4.6: Points of $f(\alpha)$ connect pieces of R and B and form a unimodal continuous function

Figure 4.7: The function $\overline{M}$ is unimodal. For interpretation of the references to color in this and all other figures, the reader is referred to the electronic version of this dissertation.

transmitted by r when it moves to $pos_r(\alpha)$ and $g_b(\alpha)$ the largest value that s_1 can transmit to $pos_r(\alpha)$. Similarly, we define $pos_b(\alpha)$ to be the point along rs_α that maximizes the value that s_1 can transmit, we denote this value by the function $h_b(\alpha)$. We also associate with $pos_b(\alpha)$ the function $h_r(\alpha)$, denoting the maximum value that r can transmit to s_2 when it moves to $pos_b(\alpha)$. Lastly, we define the function $f(\alpha)$ to be the value at the point on segment rs_α such that both s_1 and r can transmit the same amount of data to/from (when there are more than one point with this property, then $f(\alpha)$ denotes the highest value).

We observe that

$$\overline{M}(\alpha) = \max\Big\{ \min\{g_r(\alpha), g_b(\alpha)\}, \min\{h_r(\alpha), h_b(\alpha)\}, f(\alpha)\Big\} \tag{4.5}$$

We base the proof that $\overline{M}(\alpha)$ is unimodal on the following claims (which we prove in appendix A):

Claim 1. *$g_r(\alpha)$ is unimodal*

Claim 2. *$h_b(\alpha)$ is decreasing*

Claim 3. *$f(\alpha)$ is unimodal*

Claim 4. *$g_b(\alpha)$ is continuous*

Claim 5. *$h_r(\alpha)$ is continuous*

Since, $g_r(\alpha)$ maximizes the message that r can transmit, then

$$\begin{aligned} f(\alpha) &\leq g_r(\alpha) \\ h_r(\alpha) &\leq g_r(\alpha) \end{aligned} \tag{4.6}$$

Similarly, from the perspective of s_1

$$\begin{aligned} f(\alpha) &\leq h_b(\alpha) \\ g_b(\alpha) &\leq h_b(\alpha) \end{aligned} \tag{4.7}$$

We consider two cases:

1. $g_r(\alpha) < g_b(\alpha)$
2. $h_b(\alpha) < h_r(\alpha)$

When the first condition holds, (4.6) and (4.7) imply

$$\begin{aligned} & h_r(\alpha) \leq g_r(\alpha) \leq g_b(\alpha) \leq h_b(\alpha) \\ & \overline{M} = \max\Big\{ \min\{g_r(\alpha), g_b(\alpha)\}, \min\{h_r(\alpha), h_b(\alpha)\}, f(\alpha) \Big\} \\ & \quad = \max\Big\{ g_r(\alpha), h_r(\alpha), f(\alpha) \Big\} \\ & \quad = g_r(\alpha) \end{aligned}$$

So in this case, $\overline{M}(\alpha) = g_r(\alpha)$ as shown in Figure 4.3. Similarly, when the second condition holds, $\overline{M}(\alpha) = h_b(\alpha)$ as shown in Figure 4.4.

Since $g_r(\alpha)$ is unimodal and $h_b(\alpha)$ is decreasing, the function $\min\{g_r(\alpha), h_b(\alpha)$ is unimodal. Let R be the set points of $g_r(\alpha) \cap \overline{M}(\alpha)$ and B the set of points of $h_b(\alpha) \cap \overline{M}(\alpha)$. Since points of R satisfy the first condition and points of B satisfy the second condition, then

$$R \cup B \subseteq \min\{g_r(\alpha), h_b(\alpha)\}$$

and hence, $R \cup B$ form a unimodal set (Figure 4.5).

The remaining points of $\overline{M}(\alpha)$ are points satisfying $\overline{M}(\alpha) = f(\alpha)$. We know using claims 4 and 5 that the functions $g_b(\alpha)$ and $h_r(\alpha)$ are continuous. So the endpoint of each interval in R and B is also a point on $f(\alpha)$. On the other hand, $f(\alpha)$ forms a unimodal set (by claim 3). Therefore, the function $\overline{M}(\alpha)$ is continuous and unimodal as Figure 4.6 illustrates. □

Load Balancing Heuristic

In this section, we propose a simple heuristic for solving the base case of 1-MMRC. This heuristic is faster than the optimal algorithm proposed above and achieves comparable data gathering capacities. This heuristic increases the data gathering capacity of the base case by balancing the energy consumption loads between the transmitting node s_1 and the mobile relay r proportionally according to their available energy. The main idea is to consider only points on the line (s_1s_2) as target positions for r, find an approximate position r' on (s_1s_2) and then refine r' into the final position r''. We compute our solution as follows. First, we set r' as the midpoint of s_1s_2. We then compute the amount of energy e_m, consumed by r when it moves from its original position to r'. This amount corresponds to an estimate of the amount of energy consumed when r moves to its final position r''. We use $e'' = e_r - e_m$ as an estimate of the amount of energy available at r after it moves to r''. Finally, we use e_1 and e'' to compute the position r'' that maximizes the data gathering capacity of s_1, r, and s_2. Let L be the length of the segment (s_1s_2) and d the distance between s_1 and r'', the optimal d satisfies:

$$\frac{e_1}{a_1 + b_1 d^2} = \frac{e''}{a_r + b_r(L-d)^2} \Leftrightarrow$$

$$(e_1 b_r - r'' b1)d^2 - 2r_1 b_r L d + e_1(a_r + b_r L^2) - e'' a_1 = 0 \tag{4.8}$$

We solve the quadratic equation (4.8) to obtain the optimal value d and then compute r'' as $r'' = s_1 + \frac{d}{L}(s_2 - s_1)$.

4.2.2 The Assignment Problem

Because of our assumption that each transmission link can be helped by at most one relay node, our objective is to find a matching between mobile relay nodes and transmission links in the underlying transmission topology. Thus, all of our centralized algorithms employ a matching or assignment algorithm as a key step. It is possible that some mobile relay nodes may not be used in the matching, but this is unlikely in practice because the number of relay nodes will typically be much smaller than the number of static nodes. We also must find the optimal position of each mobile relay node.

With our optimal base case algorithm, we now know where each relay node r should move if we know which transmission link r should help; that is, we need to match each mobile relay with a transmission link such that the application objective is optimized. This is essentially the assignment problem from combinatorial optimization where we typically have n tasks and n people, each task needs to be assigned to a single person, and the cost or efficiency of the i-th person performing the j-th task is given by c_{ij}. The goal is to assign people to tasks in order to maximize efficiency or minimize costs. The assignment problem is often formalized as a 0-1 integer program where the 0-1 variable x_{ij} takes on value 1 if person i is assigned to task j and 0 otherwise. Fortunately, we can relax this integer program to a linear program where x_{ij} can take on fractional values because the constraint matrix of the linear program is totally unimodular which means that an integral optimal solution is guaranteed. The linear programming formulation is the following:

$$\begin{cases} \max \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \\ \sum_{j=1}^{n} x_{ij} = 1 \quad (i = 1, 2, \ldots, n) \\ \sum_{i=1}^{n} x_{ij} = 1 \quad (j = 1, 2, \ldots, n) \\ x_{ij} \in \{0, 1\}, \quad (i = 1, , \ldots, n, \quad j = 1, \ldots, n) \end{cases}$$

The first constraint guarantees that each person is assigned to one task, and the second constraint guarantees that each task is assigned to one person. The objective function attempts to maximize the sum of task efficiencies and can be solved in $O(n^3)$ time [8]. We use this formulation for 1-MMRC in a star. We can create the maximum bottleneck assignment problem by changing the objective function from $\max \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$ to $\max \min_{1 \leq j \leq n} \sum_{i=1}^{n} c_{ij} x_{ij}$. This formulation can be solved in $O(n^{2.5})$ time [8]; we use it for 1-MMRC in a line and for 1-MMRC in a tree with data reduction.

4.2.3 1-MMRC in a Line, Star, and Tree with Data Reduction

Consider an arbitrary 1-MMRC in a line instance. Let r be the number of mobile relay nodes, k be the number of static nodes on the transmission path, and $n = r + k$ be the total number of nodes. Let $P = (s_1, \ldots, s_k)$ be the path of static nodes where s_1 is the source node and s_k is the sink node.

The basic idea in transforming this 1-MMRC instance to a maximum bottleneck assignment problem instance is the following. The relay nodes take the place of people. The transmission links $l_j = (s_j, s_{j+1})$ take the role of jobs. To allow for the case where no relay nodes are used, we add $k-1$ trivial "people" d_j to represent the case when node s_j transmits directly to s_{j+1} and r trivial jobs out_i to represent the case where relay node r_i is not used.

For actual relays and transmission links, we use the base case algorithm in Section 4.2.1 to compute c_{ij}, the size of the largest data chunk that can be transmitted from s_j to r_i to s_{j+1}. We can alternatively use the approximation of Section 4.2.1 to obtain a faster close to optimal value. If relay node r_i cannot increase the size of data that s_j can transmit, then we set c_{ij} to 0. Let c_j be the size of data if the transmission is done directly from s_j to s_{j+1}. The efficiency matrix values would then be:

	r_1	...	r_r	d_1	...	d_{k-1}
l_1	c_{11}	...	c_{r1}	c_1	...	0
l_2	c_{12}	...	c_{r2}	0	...	0
	...	...			...	
l_{k-1}	c_{1k-1}	...	c_{rk-1}	0	...	c_{k-1}
out_1	∞	...	∞	∞	...	∞
	...	...			...	
out_r	∞	...	∞	∞	...	∞

More formally,

$$A(l_j, r_i) = c_{ij} \qquad A(out_j, r_i) = \infty$$
$$A(l_j, d_j) = c_j \qquad A(out_j, d_i) = \infty$$
$$A(l_j, d_i) = 0 \text{ if } i \neq j$$

We use the exact same formulation for 1-MMRC in a Tree with Data Reduction. For 1-MMRC in a Star, we change the objective function to $\max \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$.

4.2.4 1-MMRC in a Tree without Data Reduction

The assignment based solution we described in Section 4.2.2 can not be applied to this 1-MMRC variant since the LP formulation does not capture the correct bottleneck edge which is no longer the edge with the smallest capacity; adding more constraints to solve this issue breaks the unimodularity property of the constraint matrix and subsequently the

integer program can not be solved using linear programming techniques. Instead, we solve a decision version of this problem using unweighted bipartite matching algorithms. For the decision version, we add an input M and the yes/no question is whether or not there exists an assignment of mobile relay nodes to transmission links such that every source node is able to send M events to the sink. We then converge on an optimal solution by using standard doubling and binary search techniques. We make an initial guess M and keep doubling this guess until it is infeasible. We then do a binary search between the last feasible value and the first infeasible value. If the initial M was not feasible, we binary search between 0 and M.

To solve the decision version, we first compute for each node the amount of data that it needs to send so that every M data units initiating at a source and passing through it can reach the sink by doing a bottom-up traversal of the tree. For each source, we set its value to M. For each internal node, we set its value to be the sum of the values of its children. Let M_i be the value that node s_i needs to transmit. Second, we eliminate any node that can send M_i bits without help. Third, we create a bipartite graph G_b as follows. For each remaining static node s_i, we create an edge between s_i and relay node r_j if r_j can help s_i reach its M_i value. Finally, we search for a bipartite matching in G_b. If there is a bipartite matching such that all nodes in G_b are matched with some relay node, we return yes. Otherwise, we return no. Bipartite matchings can be found in $O(n^{2.5})$ time [21].

4.2.5 Distributed Algorithm

Before describing our distributed implementation, we propose three centralized greedy algorithms that provide insight to the distributed approach; these greedy strategies differ

```
procedure staticRun
    ▷ Initialize empty priority queue
    offers ← ∅;
    committed ← false;
    repeat
        ▷ Listen to mobile nodes in range and collect offers presented
        repeat
            receive(mobile, type, data);
            if type = MOBILE_IN_RANGE then
                send(mobile, META_DATA, info);
            else if type = OFFER then
                offers.add(data);
            end if
        until timeout

        ▷ Process offers and pick best
        if offers ≠ ∅ then
            bestOffer ← offers.dequeue();
            send(bestOffer.sender, ACCEPT_OFFER);
            committed ← true;
        end if
        while offers ≠ ∅ do
            offer ← candidates.dequeue();
            send(offer.sender, REJECT_OFFER);
        end while
    until committed
end procedure
```

Figure 4.8: Local algorithm executed by each static node

only in their link priority strategies which determine which link to process first. The three algorithms proceed as follows. First, for each link l between two static nodes s_i and s_j, we compute the maximum amount of data $c(l)$ that s_i can transmit to s_j. Second, for each mobile node r and for each link l, we compute the total amount of data $c(r, l)$ that can be transmitted from s_i to r to s_j using the optimal algorithm for the base case (section 4.2.1). If s_i can send more data directly to s_j than going through r, we set $c(r, l)$ to zero. We now choose the link with the highest priority using three different priority schemes: (1) the highest value $c(r, l)$, (2) the largest improvement $c(r, l) - c(l)$ and (3) the smallest (bottleneck) direct value $c(l)$. In case of a tie for the third metric, we pick the link with the

```
procedure MOBILERUN
    ▷ Initialize empty priority queue
    candidates ← ∅;
    committed ← false;
    repeat
        BROADCAST(MOBILE_IN_RANGE);

        ▷ Collect responses from static nodes in range
        repeat
            RECEIVE(sender, META_DATA, sender.info);
             ▷ Decision data include: data that can be sent directly,
             ▷ through mobile node, position of mobile node, priority
            sender.decisionData ← COMPUTEAMOUNTS(sender.info)
            candidates.add(sender);
        until timeout

        ▷ Process candidates queue
        while candidates ≠ ∅ do
            candidate ← candidates.dequeue();
            SEND(candidate, OFFER, data(candidate, self));
            response ← RECEIVE(candidate, responseType);
            if responseType = ACCEPT_OFFER then
                committed ← true;
                break;
            end if
        end while
    until committed
    MOVETO(candidate.optimalPosition);
end procedure
```

Figure 4.9: Local algorithm executed by each mobile node

highest $c(r, l)$. In all three cases, we match the selected link to the corresponding mobile relay r and then repeat the greedy selection process until all the mobile nodes are matched to links or cannot help the remaining links.

We now propose a distributed approach that mimics our greedy algorithms. For MMRC in a line or tree, we use the bottleneck link priority scheme. For MMRC in a star, we use the largest improvement link priority scheme. The algorithm begins with each mobile relay node broadcasting its existence (a MOBILE_IN_RANGE message) to nodes within its neighborhood. We define the neighborhood of a mobile node r as the set of nodes that can

be reached within h hops from r for a given parameter h. Each static node that receives such a message responds by sending its location, its remaining energy, and the location of its destination static node (a META_DATA message) to each mobile relay node that sends it a broadcast message. Each mobile relay node r then calculates for each responding static node s how much that s can transmit now and how much r can improve s's transmission capacity. It then sends a message (of type OFFER) to the static relay node with the highest priority saying how much it can help it and waits for a response from that static node. Each static node that receives an offer accepts the best offer it receives, sending an acceptance message to the one relay node it accepts and rejection messages to the other relay nodes. Rejected relay nodes then send messages to the next node on their priority list. If a static node that is already committed receives a late offer from a relay node, it automatically rejects the offer even if that relay node can provide more help. We show the pseudocode for the local algorithm performed by the static and mobile nodes in Figures 4.8 and 4.9 respectively.

4.3 2-MMRC: Two Relays per Link

The previous algorithms are optimal or close to optimal under the constraint that a single mobile relay node can help at most a single link. In some cases, the network contains a large number of mobile relays, or long transmission links. These cases may benefit from further decomposing the transmission links into multiple smaller links, to exploit the availability of relay nodes and distribute the energy consumption over several nodes. In this section, we relax the above constraint and study the benefit of allowing two relays to help a single static link. We first consider the modified base case when two relay nodes are available in the proximity of a link. Then we extend our distributed algorithm of Section 4.2.5 to take into

account two relays joining a link.

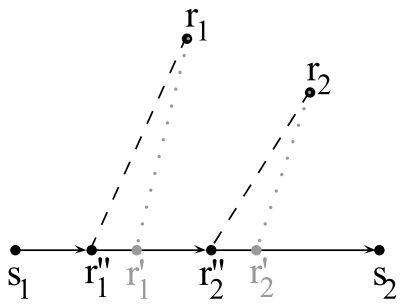

Figure 4.10: Using r_1 and r_2 at r_1'' and r_2'' increases the amount of data that can be transmitted from s_1 to s_2.

4.3.1 Base Case

We consider the base case of 2-MMRC as shown in Figure 4.10. Similar to the base case of Section 4.2.1, we would like to find the optimal positions of r_1 and r_2 such that the amount of data that can be sent from s_1 to s_2 through r_1 and r_2 is maximized. A direct analytical solution is equivalent to solving the following nonlinear program which is a complex and challenging task.

$$\underset{x_{r_1},y_{r_1},x_{r_2},y_{r_2}}{\text{Maximize}} \quad m$$

subject to

$$\frac{e_1}{a_1 + b_1\left((p_1 - x_{r_1})^2 + (q_1 - y_{r_1})^2\right)} \geq m$$

$$\frac{e_{r_1} - k_{r_1}\sqrt{(x_{r_1} - p_{r_1})^2 + (y_{r_1} - q_{r_1})^2}}{a_{r_1} + b_{r_1}\left((x_{r_1} - x_{r_2})^2 + (y_{r_1} - y_{r_2})^2\right)} \geq m$$

$$\frac{e_{r_2} - k_{r_2}\sqrt{(x_{r_2} - p_{r_2})^2 + (y_{r_2} - q_{r_2})^2}}{a_{r_2} + b_{r_2}\left((x_{r_2} - p_2)^2 + (y_{r_2} - q_2)^2\right)} \geq m$$

We observe that in optimal solutions, the final locations for r_1 and r_2 are close to the line connecting s_1 and s_2 even when the mobility parameter k is high. This is because when two mobile relays join the transmission line, the corresponding transmission distances are much smaller than the original direct transmission distance between s_1 and s_2. Since the energy consumption due to transmissions is quadratic in the transmission distance, the savings in transmission energy are high and make up for a relatively long movement for the mobile nodes.

Consistent with this observation, we extend the load balancing heuristic of 1-MMRC to solve the base case of 2-MMRC. Again, our new load balancing heuristic only considers the line (s_1s_2) for target positions of r_1 and r_2. This constraint reduces the complexity of the computation without noticeable loss in the performance as we show in Section 5.6. The target positions for r_1 and r_2 are computed as follows. First, we estimate the amount of energy that r_1 and r_2 consume to get to their target positions on s_1s_2 by picking two estimate destination points on that line, r_1' and r_2' respectively (shown in Figure 4.10). These points are computed as the evenly spaced points along (s_1s_2) as follows:

$$r_1' = \frac{2}{3}s_1 + \frac{1}{3}s_2 \quad r_2' = \frac{1}{3}s_1 + \frac{2}{3}s_2$$

Then, we use these positions to estimate the remaining energies e'_{r_1} and e'_{r_2} at r_1 and r_2 when they relocate to r_1' and r_2'. Finally, we compute the final destination positions r_1'' and r_2'' as the points at which nodes s_1, r_1 and r_2 consume their remaining energy at the same rate. We solve for the values d_1 (distance between s_1 and r_1'') and d_2 (distance between r_1''

and r_2'') by finding the values satisfying the following equation:

$$\frac{e_1}{a_1 + b_1 d_1^2} = \frac{e'_{r_1}}{a_{r_1} + b_{r_1} d_2^2} = \frac{e'_{r_2}}{a_{r_2} + b_{r_2}(L - d_1 - d_2)^2} \tag{4.9}$$

where L is the length of the segment $s_1 s_2$. Equation (4.9) is equivalent to

$$d_2^2 = \frac{e'_{r_1} a_1 - e_1 a_{r_1}}{e_1 b_{r_1}} + \frac{e'_{r_1} b_1}{e_1 b_{r_1}} d_1^2 \tag{4.10}$$

$$(L - d_1 - d_2)^2 = \frac{e'_{r_2} a_1 - e_1 a_{r_2}}{e_1 b_{r_2}} + \frac{e'_{r_2} b_1}{e_1 b_{r_2}} d_1^2 \tag{4.11}$$

(4.10) and (4.11) imply that d_1 satisfies

$$\begin{aligned} &((\beta_2 - \beta_1 - 1)^2 - 4\beta_1) d_1^4 + (4L\beta_1 - 4L + 4L\beta_2) d_1^3 \\ &\quad + (-4L^2\beta_1 - 4\alpha_1 + 2(\alpha_2 - \alpha_1 - L^2)(\beta_2 - \beta_1 - 1) + 4L^2) d_1^2 \\ &\quad + (4L\alpha_2 + 4L\alpha_1 - 4L^3) d_1 \\ &\quad + (\alpha_2 - \alpha_1 - L^2)^2 - 4L^2\alpha_1 = 0 \end{aligned} \tag{4.12}$$

which is a quartic polynomial whose roots can be found efficiently using Ferrari's method [54] for quartic equations. Once d_1 is computed, d_2 and consequently r_1'' and r_2'' can be found in a straightforward manner.

4.3.2 Distributed Algorithm

The transformation into an assignment problem described in Section 4.2.3 does not apply in this case because it allows for a mobile relay node to be used multiple times, each time coupled with a different relay. Adding a constraint to the linear program to ensure this does

not happen breaks the unimodularity property of the linear program and consequently the guarantee of an integral solution.

Based on the distributed implementation of Section 4.2.5, we propose a greedy algorithm to match mobile relays to links as follows. The main idea is to consider the critical links first. Each such link considers the surrounding mobile nodes that did not join the tree, both individually and in pairs. When r is considered by itself by a link $s_i s_j$, the optimal position for the single relay base case (Section 4.2.1) is used to compute the corresponding data gathering capacity of the subpath $\langle s_i, r, s_j \rangle$. Likewise, when two mobile relays r_1 and r_2 are considered, the solution described in Section 4.3.1 is used to compute the data gathering capacity of $\langle s_i, r_1, r_2, s_j \rangle$. Node s_i then picks the relay node or relay pair that results in the highest priority. We note that we allow two relay nodes to join a single link only if the expected priority is greater than the highest priority of a single relay for that link by a factor f. This is to ensure the improvement is significant when two relay nodes are used for one link. This process is repeated until all the mobile relays are assigned to some link or the available mobile relays cannot benefit the links by joining the transmission lines.

4.4 Simulation Results

4.4.1 Setup

We evaluated our algorithms using simulations. For a given simulated network, we varied the number of mobile nodes by 5 between 5 and 30. For the star and tree topologies, we varied the number of source nodes between 5 and 30 by 5. For each number of source nodes and number of mobile relay nodes, we generated 20 random networks (120 for the

line, 720 for the tree and star) consisting of 100 nodes in a 150m by 150m area. To ensure high quality transmissions based on empirical data using Tmotes [45], we set the maximum communication distance to 35m. We set the initial battery energy of nodes uniformly at random between half charged and fully charged. We modeled different platforms and different settings by having 16 possible choices for node energy consumption parameters (4 values for a and b and 4 values for k). To build the communication trees between the sources and sink, we used greedy geographic forwarding in which each node forwards the packet to the neighbor closest to the sink. We note that our solutions only require the establishment of a routing topology before mobile relay configuration, and hence can work with other routing algorithms as well.

For all networks that we generated, we evaluated our optimal algorithm as well as seven different greedy strategies. Six of our greedy strategies differed in two main ways: point selection and link priority. We consider two different point selection strategies. The first is to move the relay node to the midpoint of the two static nodes defining the link that will be helped; we label such greedy strategies with the suffix -*MP*. The other option is to compute the optimal position using our optimal base case algorithm; these strategies are labeled with the suffix -*OPT*. We consider the three link priority schemes we described in the distributed implementation section: the highest value, the largest improvement, and the bottleneck value with ties broken by largest improvement. The net result is six greedy strategies. We also consider a seventh greedy algorithm that uses matching to minimize the total travel distance of all mobile relay nodes. Finally, we evaluated our distributed implementations, one for each of the three link priority strategies. To model the asynchronicity of our protocol, we vary the fraction of mobile relay nodes that participate in the protocol at various percentages between

25% and 100%. We also vary the neighborhood parameter h between $1, 2, 3$ and unlimited. We did not implement existing algorithms because their fundamental assumptions do not match ours making meaningful comparisons difficult if not impossible.

We also generated a new set of topologies, consisting of 100 base cases to assess our optimal algorithm and load balancing heuristics for solving the base case. Each network in this set consists of a single static link and two mobile relays. When assessing the base case for 1-MMRC, only one of the existing mobile relays (the more beneficial one) is allowed to join the link whereas in 2-MMRC solutions, both relay nodes are expected to join.

For a network N and an algorithm A, let $A(N)$ be the data gathering capacity of the resulting network N' computed by algorithm A. Let B denote the baseline algorithm that uses no mobile relay nodes; thus, the resulting network N' is the original network of static nodes. We measure the performance of algorithm A on network N by its improvement ratio which is $A(N)/B(N)$. When reporting results for an algorithm A, we report the average of $A(N)/B(N)$ for all N that have the same number of mobile relay nodes. For line topologies, this will be an average of 20 networks; for tree and star topologies, this will be an average of 120 networks (20 for each of 6 values of the number of source nodes). We also report results for where we average over all networks for the given application. We now present a summary of our simulation results for each application.

4.4.2 1-MMRC in a Line

We highlight some of our key results for MMRC in a Line in Figure 4.11. The average improvement ratio for our optimal algorithm over all 120 networks is 232%, and the improvement ratio increases from 180% to 285% as we increase the number of mobile relay nodes.

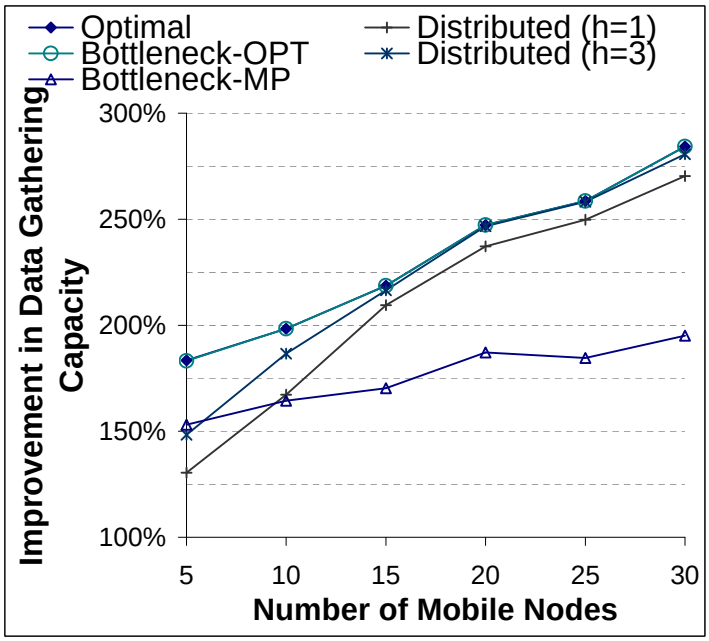

Figure 4.11: Improvement in data gathering capacity of our algorithms in line topologies

The best greedy strategy, Bottleneck-OPT, also has an average improvement ratio of 232%. As we can see from Figure 4.11, its improvement ratio is very close to that of the optimal algorithm for all numbers of mobile relay nodes. This implies that we can use a simple greedy strategy to perform the matching rather than using more sophisticated weighted matching algorithms. The remaining greedy strategies perform signficantly worse with average improvement ratios ranging from 165% to 210%. We also observe the importance of optimal point selection; the Bottleneck-MP greedy strategy achieves an average improvement ratio of 175% which is much worse than that of Bottleneck-OPT. In fact, for each link priority scheme, the midpoint point selection strategy has a significantly lower improvement ratio than its optimal point selection counterpart.

Our distributed implementation of the Bottleneck-OPT greedy strategy has an average improvement ratio of 211% when the neighborhood parameter $h = 1$. This ratio is a bit lower than both the optimal algorithm and Bottleneck-OPT. There are two possible explanations

for the slight decrease in performance. The first is that a relay node may not be matched to the correct link because it cannot reach the source node of the link with its transmission (*e.g.* it is farther than 35m away from the source node of the link); the second is that a relay node may not be matched to the correct link because its offer of help arrives after the source node of the link has already accepted the help of another relay node. Our results indicate that the 35m transmission distance limitation is the more important factor, particularly as the number of relay nodes increases. When h increases to 3, the improvement ratio increases by 12% to an average of 223%. We also observe that it is enough to set h to 3 to get the maximum improvement for the distributed approach. Moreover, we see little difference between the distributed implementation with 33% of the relay nodes participating and 100% of the relay nodes participating for any number of relay nodes; this also holds for all values between 33% and 100%. We also see that the distributed implementation's average improvement ratio approaches that of the optimal algorithm as the number of relay nodes increases. In this case, each transmission link is likely to receive an offer from a good if not optimal relay node.

4.4.3 1-MMRC in a Star

We highlight some of our key results for 1-MMRC in a Star in Figure 4.12. The average improvement ratio for our optimal algorithm over all 720 networks is 155%, and the improvement ratio increases from 130% to 170% as we increase the number of mobile relay nodes. The best greedy strategy, Improvement-OPT, has an average improvement ratio of 152% which is nearly identical to the optimal algorithm for all numbers of relay nodes. The remaining greedy strategies perform significantly worse with improvement ratios ap-

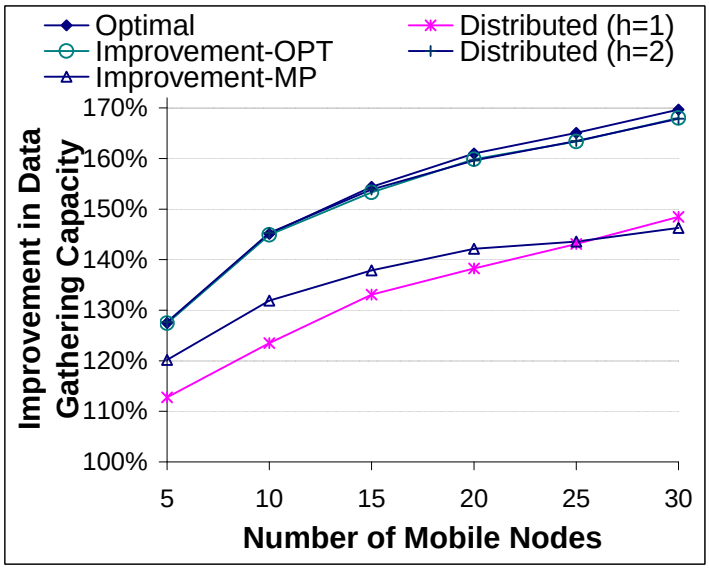

Figure 4.12: Improvement in data gathering capacity of our algorithms in star topologies

proximately 20% lower. We again observe the importance of optimal point selection; all the greedy strategies that use optimal point selection have improvement ratios roughly 17% better than their midpoint point selection counterparts.

Our distributed implementation of the Improvement-OPT greedy strategy has an improvement ratio of 133% for $h = 1$ and 152% for $h = 2$. We note that setting h to 2 allows each mobile node to consider all the links in the network and consequently the improvement ratios are very close to the optimal ratios. We again observe that the main cause of the performance loss is the 35m transmission distance limit as our distributed algorithms perform essentially identically once at least 33% of the relay nodes participate in the protocol. We again see the distributed implementation's performance increase significantly with the number of mobile relay nodes.

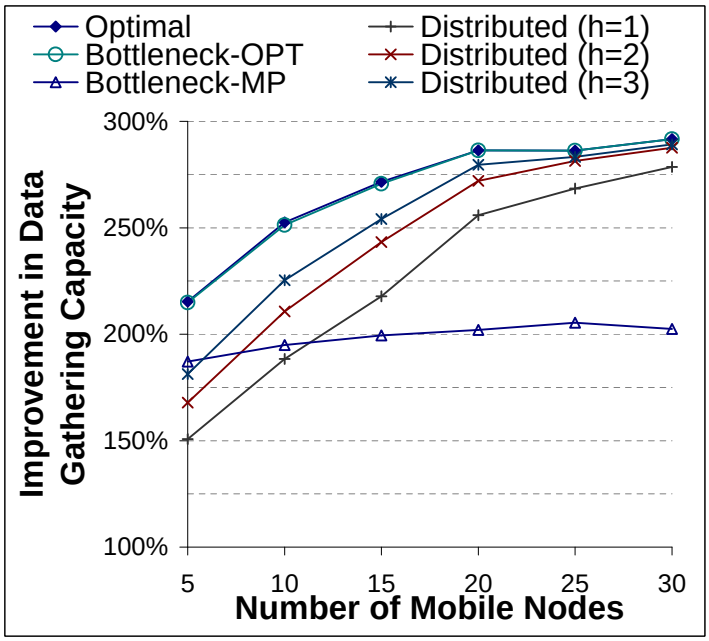

Figure 4.13: Improvement in data gathering capacity of our algorithms in tree topologies with data reduction at the nodes

4.4.4 Fair 1-MMRC in Trees with Data Reduction

We highlight some of our key results for 1-MMRC in trees with data reduction in Figure 4.13. The average improvement ratio for our optimal algorithm over all 720 networks is 270%, and the improvement ratio increases from 215% to 300% as the number of mobile relays increases. The best greedy strategy, Bottleneck-OPT, has an average improvement ratio of 268% which is nearly identical to the optimal algorithm in all cases. The remaining greedy strategies perform significantly worse with average improvement ratios of at most 240%. We again observe the importance of optimal point selection; the greedy strategies that use optimal point selection have improvement ratios roughly 65% higher than their midpoint point selection counterparts. For Bottleneck-OPT and Bottleneck-MP, the divergence in performance increases as the number of mobile relays increases.

We now evaluate the performance of our distributed implementation for different values

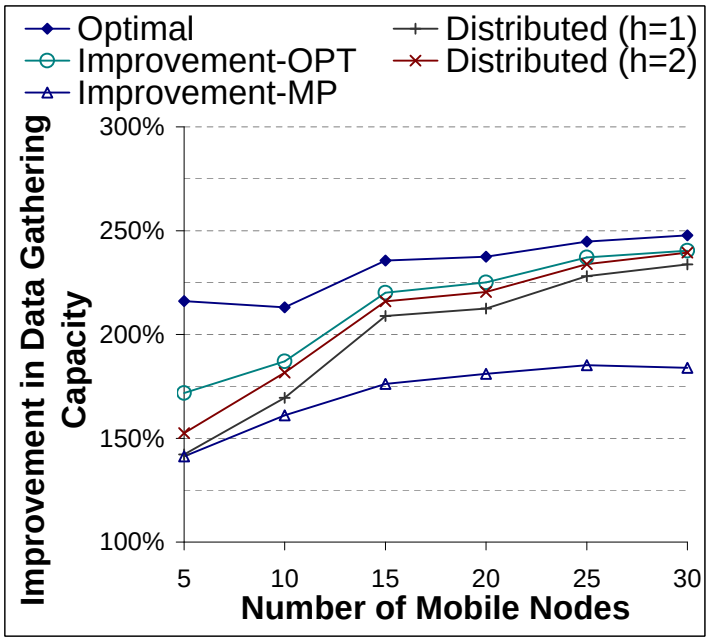

Figure 4.14: Improvement in data gathering capacity of our algorithms in tree topologies without data reduction at the nodes

of h. We obtain improvement ratios of 226%, 242% and 252% for $h = 1, 2$ and 3 respectively. We also obtain the same improvement ratios for all $h \geq 3$. We again observe that the main cause of the performance loss is the 35m transmission distance limit since the improvement ratios increase as h increases. Similar to the previous results, our distributed algorithms perform essentially identically once at least 33% of the relay nodes participate in the protocol. We again see the distributed implementation's performance approaches that of the optimal algorihtm as the number of mobile nodes increases. We also study how many communication rounds are required before all the mobile nodes commit to some edge or remove themselves from consideration. In all cases, the number of rounds grows very slowly with the number of mobile nodes. On average, only 4 to 6 rounds are needed. Even when the number of relays was high (≈ 30), the number of rounds never exceeded 8. These results also hold for MMRC in trees without data reduction.

4.4.5 Fair 1-MMRC in Trees without data reduction

We highlight some of our key results for 1-MMRC in trees without data reduction in Figure 4.14. The average improvement ratio for our optimal algorithm over all 720 networks is 233%, and the improvement ratio increases from 216% to 250% as the number of mobile relay increases. The best greedy strategy, Bottleneck-OPT, has an average improvement ratio of 220% and it approaches that of the optimal algorithm as we increase the number of mobile relay nodes. Improvement-OPT performs almost as well as Bottleneck-OPT, particularly for many mobile relay nodes. The remaining greedy strategies perform significantly worse with improvement ratios of at most 175%. We again observe the importance of optimal point selection; the greedy strategies that use optimal point selection have improvement ratios roughly 40% higher than their midpoint point selection counterparts. For Bottleneck-OPT and Bottleneck-MP, the divergence in performance increases as the number of mobile nodes increases.

Our distributed implementation based on Bottleneck-OPT again performs well with average improvement ratios of 200% and 208% for $h = 1$ and 2 respectively. We note that it is enough to set h to 2 to get similar improvement ratios as the unlimited neighborhood case. When $h \geq 2$, the improvement ratios obtained are within 5% from Bottleneck-OPT when the number of mobile nodes exceeds 5. We also show that the average number of messages generated per mobile relay node grows slowly and linearly with number of mobile nodes. For example, when a static node receives offers from 33% of the mobile relay nodes in its neighborhood, the average number of messages sent by each relay node is only 2.1. Even with large number of mobiles nodes (30) and high communication ratio (100%), the number

of messages generated by each mobile node is only 4.

Finally, in our simulations for all four applications, the approach that matches mobile relays based on minimizing the total distance traveled produces noticeably inferior results to the optimal approach and, in most cases, the other greedy approaches. This shows that optimizing only to minimize movement without regard to transmission is not helpful. In particular, it is helpful to spend a little more energy on movement if this will significantly decrease energy consumed by transmissions.

4.4.6 Two Relays Per Link

We now evaluate the benefit of allowing two relay nodes to join a link. We first compare our base case algorithms for 1-MMRC and 2-MMRC. Then, we compare our distributed algorithms for general topologies.

Base Case

To evaluate our algorithms for the base case, we ran the following simulations on the set of base cases we generated. For each base case topology, we computed the optimal data gathering capacity using our optimal algorithm for 1-MMRC and exhaustive search for 2-MMRC and using our load balancing heuristics for both variants. The average improvement ratios are shown in Table 4.1.

Table 4.1: Average Improvement Ratios for Base Case Problems

	Optimal Improvement Ratio	Heuristic Improvement Ratio	Heuristic Ratio / Optimal Ratio
Single Relay	460.7%	460.5%	99.3%
Double Relay	591.2%	585.6%	98.9%

We observe that in both 1-MMRC and 2-MMRC base cases, our approximation algo-

rithms result in solutions very close to optimal, within 99% for both cases. However, in 2-MMRC base cases, the improvement in data gathering capacity is larger, reaching an average of 591% compared to 460% in 1-MMRC base cases. So, when considering base case topologies, 2-MMRC outperforms 1-MMRC.

Fair 2-MMRC in Trees with Data Reduction

We now evaluate the performance of 2-MMRC in general topologies. For this purpose, we compare the total data gathering capacity achieved by 1-MMRC and 2-MMRC in tree topologies with data reduction. For 1-MMRC, we set h to 3. For 2-MMRC, we varied h between 1 and 3 and set the improvement threshold f to 1.25. We highlight the results in Figure 4.15. We make the following observations. First, consistent with the previous results, the performance increases as h increases from 1 to 3. We note that it is enough to set h to 3 to get very close performance to the unrestricted case where each mobile relay considers the whole network. Second, for $h = 3$, 2-MMRC slightly outperforms 1-MMRC with the difference being at most 4% which is much less than the divergence observed for base cases. This is because when the number of mobile nodes in the network is limited, using at most one relay node per links allows more links in the network to be helped and consequently more bottlenecks to be improved. In 2-MMRC solutions, only a small number of relay nodes are used in pairs (≤ 3 pairs), most links are helped by a single relay node. This is due to two factors: the parameter f and the geographic distribution of mobile nodes. Aside from a few cases, most relay nodes are scattered in the network and are not close enough to each other to collectively increase the data gathering capacity of a link. We note that we get similar results for topologies without data compression at the nodes. We conclude that it is more

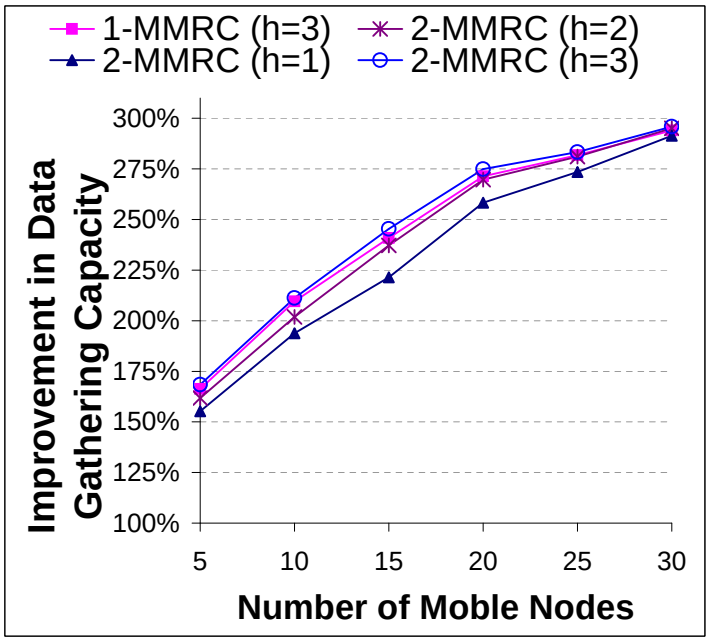

Figure 4.15: Improvement in data gathering capacity of the distributed implementations for 1-MMRC and 2-MMRC for trees with data reduction

beneficial and efficient to compute the data gathering capacity using the 1-MMRC variant of MMRC.

4.5 Summary

In this chapter, we study a new problem, MMRC, maximizing the data gathering capacity of hybrid wireless sensor networks consisting of both mobile and static nodes. Our solutions consist of inserting the mobile nodes along the transmission lines to improve the capacity of weak links. We presented optimal solutions to four variants of MMRC when a single relay is allowed per link, and approximate solutions when two relay nodes are allowed per link. Unlike most previous work that exploits controlled mobility, we consider both the energy consumed during the transmission process as well as the energy consumed by mechanical locomotion. In most variants, our optimal algorithm improved system lifetime by a factor of 2

or more. For the star topology, it increased system lifetime by a factor of 1.5. Our distributed protocols were almost as effective at improving data gathering capacity, particularly as the number of mobile relay nodes increases. Our simulations also showed that these protocols quickly converge on a final solution with little messaging overhead. Moreover, we show using simulations, that although using multiple relay nodes on a single link significantly improves the data gathering capacity of that link, it is better to use each relay node by itself on a single link to increase the number of links that can be helped and achieve a globally higher data gather capacity.

CHAPTER 5

Total Energy Minimization Using Mobile Nodes

In the previous scheme, the network contained a small number of mobile nodes relative to the network size. In some scenarios, the number of mobile nodes is relatively large and our previous solutions may not fully exploit this fact. In this chapter, we consider data-intensive networks that consist entirely of mobile nodes. We exploit the mobility of all the nodes in such networks to efficiently use the available energy. We consider the total energy used in the network as an indirect measure for the network lifetime. Specifically, we integrate the energy consumption due to both mobility and wireless transmissions into a holistic optimization framework and consider the problem of minimizing the total energy consumption of all participating nodes. We propose an approach in which mobile nodes join the transmission routes then all participating nodes collectively move closer to each other while maintaining connectivity. We show that our algorithms significantly reduce the energy consumption and in particular converge to the optimal configuration when the routing tree

is given.

5.1 Problem Definition

5.1.1 An Illustrative Example

Before formally defining the problem of total energy minimization, we first describe the main idea of our approach using a simple example. Suppose we have three nodes s_1, s_2, s_3 located at positions x_1, x_2, x_3, respectively (Fig. 5.1), such that s_2 is a mobile relay node. The objective is to minimize the total energy consumption due to both movement and transmissions. Data storage node s_1 needs to transmit a data chunk to sink s_3 through relay node s_2. One solution is to have s_1 transmit the data from x_1 to node s_2 at position x_2 and node s_2 relays it to sink s_3 at position x_3; that is, node s_2 does not move. Another solution, which takes advantage of s_2's mobility, is to move s_2 to the midpoint of the segment x_1x_3, which is suggested in [17]. This will reduce the transmission energy by reducing the distances separating the nodes. However, moving relay node s_2 also consumes energy. We assume the following parameters for the energy models: $k = 2, a = 0.6 \times 10^{-7}, b = 4 \times 10^{-10}$.

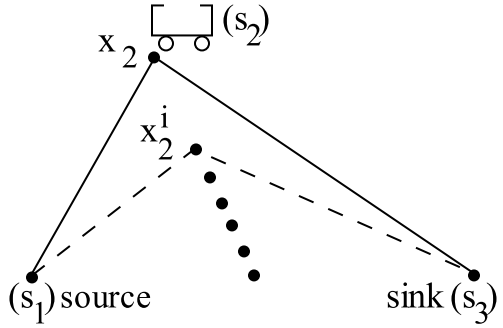

Figure 5.1: Reduction in energy consumption due to mobile relay. As the data chunk size increases, the optimal position converges to the midpoint of s_1s_3.

In this example, for a given data chunk m_i, the optimal solution is to move s_2 to x_2^i (a

position that we can compute precisely). This will minimize the total energy consumption due to both transmission and mobility. For small messages, s_2 moves very little if at all. As the size of the data increases, relay node s_2 moves closer to the midpoint. In this example, it is beneficial to move when the message size exceeds 4 MB. We illustrate in Table 5.1 the energy savings achieved using our optimal approach and the other two approaches for the relevant range of data sizes. For large enough data chunks ($\approx$ 13 MB), one relay node can reduce total energy consumption by 10% compared to the other two approaches. As the data chunk size increases further, the energy savings decrease, and the optimal position converges to the midpoint when the data size exceeds 43 MB. In general, the reduction in energy consumption is higher when there are multiple mobile relay nodes.

Table 5.1: Energy consumption comparison

Data Size (MB)	Costs at Original Pos.	Costs at Midpoints	Costs at Optimal Pos.	Reduction
5.00	42.78	70.71	42.04	1.73%
11.00	94.12	101.93	88.39	6.09%
12.00	102.68	107.13	94.71	7.75%
13.00	111.23	112.33	100.87	9.32%
14.00	119.79	117.53	106.89	9.06%
15.00	128.35	122.74	112.80	8.09%
16.00	136.90	127.94	118.62	7.28%
17.00	145.46	133.14	124.37	6.58%
18.00	154.01	138.34	130.06	5.98%
40.00	342.26	252.77	247.58	2.05%

The above example illustrates two interesting results. The optimal position of a mobile relay is not the midpoint between the source and sink when both mobility and transmissions costs are taken into consideration. This is in contrast to the conclusion of several previous studies [17, 62] which only account for transmission costs. Second, the optimal position of a mobile relay depends on not only the network topology (e.g., the initial positions of nodes) but also the amount of data to be transmitted. Moreover, as the data chunk size increases,

the optimal position converges to the midpoint of s_1 and s_3. These results are particularly important for minimizing the energy cost of data-intensive WSNs as the traffic load of such networks varies significantly with the sampling rates of nodes and network density.

5.1.2 Problem Formulation

In our definitions, we assume that all movements are completed before any transmissions begin. We also assume there are no obstacles that affect mobility or transmissions. In this case, as we show in Section 5.2.2, the distance moved by a mobile relay is no more than the distance between its starting position and its corresponding position in the evenly spaced configuration which often leads to a short delay in mobile relay relocation. We focus on the case where all nodes are in a 2-dimensional plane $\Re^2$, but the results apply to $\Re^3$ and other metric spaces.

Our problem can be described as follows. Given a network containing one or more static source nodes that store data gathered by other nodes, a number of mobile relay nodes and a static sink, we want to find a directed routing tree from the sources to the sink as well as the optimal positions of the mobile nodes in the tree in order to minimize the total energy consumed by transmitting data from the source(s) to the sink and the energy consumed by relocating the mobile relays. The source nodes in our problem formulation serve as *storage points* which cache the data gathered by other nodes and periodically transmit to the sink, in response to user queries. Such a network architecture is consistent with the design of storage-centric sensor networks [42]. Our problem formulation also considers the initial positions of nodes and the amount of data that needs to be transmitted from each storage node to the sink. The formal definition of the problem is given below.

Definition 1. *(Optimal Mobile Relay Configuration):*

Input Instance: S, a list of n nodes $(s_1, \ldots, s_n)$ in the network; O, a list of n locations $(o_1, \ldots, o_n)$ where o_i is the initial position of node s_i for $1 \leq i \leq n$; $S_{sources}$, a subset of S representing the source nodes; r, a node in S, representing the single sink; $M_{sources} = \{M_i \mid s_i \in S_{sources}\}$, a set of data chunk sizes for all sources in $S_{sources}$;

We define m_i, which we compute later, to be the weight of node s_i which is equal to the total number of bits to be transmitted by node s_i. We define a configuration $\langle E, U \rangle$ as a pair of two sets: E, a set of directed arcs (s_i, s_j) that represent the directed tree in which all sources are leaves and the sink is the root and U, a list of locations $(u_1, \ldots, u_n)$ where u_i is the transmission position for node s_i for $1 \leq i \leq n$. The cost of a configuration $\langle E, U \rangle$ is given by:

$$c(\langle E, U \rangle) = \sum_{(s_i, s_j) \in E} am_i + b\|u_i - u_j\|^2 m_i + k\|o_i - u_i\|$$

Output: $\langle E, U \rangle$, an optimal configuration that minimizes the cost $c(\langle E, U \rangle)$.

5.2 Centralized Solution

5.2.1 Energy Optimization Framework

The Optimal Mobile Relay Configuration (OMRC) problem is challenging because of the dependence of the solution on multiple factors such as the routing tree topology and the amount of data transferred through each link. For example, when transferring little data, the optimal configuration is to use only some relay nodes at their original positions. As the amount of data transferred increases, three changes occur: the topology may change by adding new relay nodes, the topology may change by changing which edges are used, and

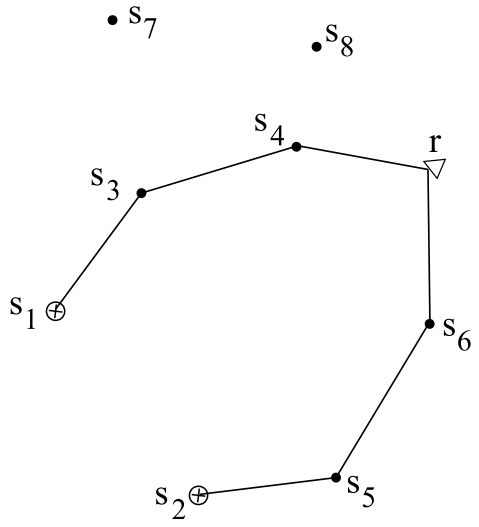

Figure 5.2: Optimal routing tree for $0 \leq m \leq 15$MB.

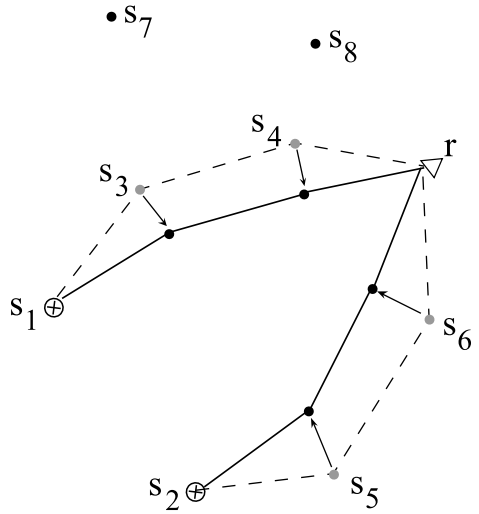

Figure 5.3: Optimal configuration for 15MB$\leq m \leq$ 25MB: same topology but nodes relocate.

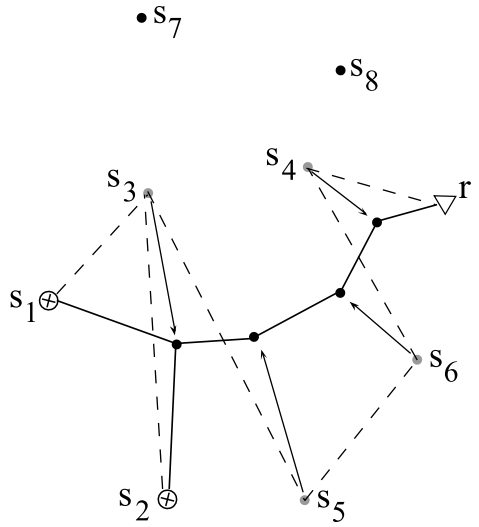

Figure 5.4: Optimal configuration for 25MB$\leq m \leq$ 60 MB. A new topology is used.

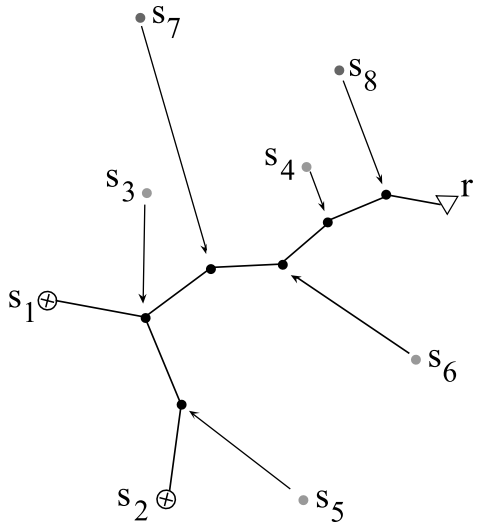

Figure 5.5: Optimal configuration for 100MB$\leq m \leq$ 150MB: a new topology with more nodes.

Figure 5.6: Example of optimal configurations as a function of amount of data to be transferred. In each part, source nodes s_1 and s_2 must send m bits of data to the sink r. We consider m up to 150MB.

the relay nodes may move closer together. In many cases, we may have restrictions such as no mobility for certain relay nodes or we must use a fixed routing tree. These constraints affect the optimal configuration.

We illustrate how the optimal configuration depends on the amount of data to transfer using the example from Fig. 5.2. When there is very little data to transfer, the optimal routing tree T_a depicted in Fig. 5.2 uses only some of the relay nodes in their original positions. When the amount of data to transfer from s_1 and s_2 increases to 15 MB, the relay nodes in tree T_a move to their corresponding positions in tree T_b of Fig. 5.3 but the topology does not change. When the amount of data to transfer from s_1 and s_2 is between 25 and 60 MB, the optimal routing tree has a different topology as shown in Fig. 5.4. For even larger messages, new trees with even more nodes included are optimal. For example, when the amount of data to be transferred is between 100 and 150 MB, the optimal tree is depicted in Fig. 5.5. No existing tree construction strategy handles all these cases. For example, the minimum spanning tree that includes all network nodes has two fundamental problems. It will typically include unneeded nodes, and it typically creates non-optimal topologies as it focuses only on the current location of nodes as opposed to where nodes may move to.

We now present a centralized approach to solve OMRC that breaks the problem into three distinct steps: initial tree construction, node insertions, and tree optimization. For each step, we present an algorithm to solve the corresponding subproblem. Our algorithm for initial tree construction is optimal for the static environment where nodes cannot move. However, we can effectively apply the later algorithms if we must start with a different topology. Our greedy heuristic for improving the routing tree topology by adding nodes exploits the mobility of the newly added nodes. Our tree optimization algorithm improves the routing

tree by relocating its nodes without changing its topology. This iterative algorithm converges on the optimal position for each node given the constraint that the routing tree topology is fixed. Our node insertion and tree optimization algorithms use the *LocalPos* algorithm we propose in Fig. 5.7 that optimally solves the simplest case (see Section 5.2.2) of the mobile relay configuration problem where there is a single source, a single sink, and a single relay node. Our approach is not guaranteed to produce an optimal configuration because we do not necessarily find the optimal topology, but our simulation results show that it performs well.

5.2.2 Base Case

Before presenting our algorithm for OMRC, we revisit the example of Section 5.1.1 as it represents the simplest possible base case of the problem in which the network consists of one source s_{i-1}, one mobile relay node s_i and one sink s_{i+1}. In this section, we calculate the optimal position for the relay node. We use the following notation. In $\Re^2$, let the original position of a node s_j be $o_j = (p_j, q_j)$, and let $u_j = (x_j, y_j)$ its final position in configuration U. According to our energy models, the total transmission and movement energy cost incurred by the mobile relay node s_i is

$$c_i(U) = k\|u_i - o_i\| + am + b\|u_{i+1} - u_i\|^2 m$$

We also define

$$C_i(U) = c_i(U) + am + b\|u_i - u_{i-1}\|^2 m$$

This corresponds to the transmission cost of node s_{i-1} plus the total cost of node s_i, which is the total cost of the final configuration in this example. We need to compute a position u_i for s_i that minimizes $C_i(U)$ assuming that $u_{i-1} = o_{i-1}$ and $u_{i+1} = o_{i+1}$; that is, node s_i's neighbors remain at the same positions in the final configuration U. We calculate position $u_i = (x_i, y_i)$ for node s_i by finding the values for x_i and y_i where the partial derivatives of the cost function $C_i(U)$ with respect to x_i and y_i become zero. Position u_i will be toward the midpoint of positions u_{i-1} and u_{i+1}. The partial derivatives $\frac{\delta C_i(U)}{\delta x_i}, \frac{\delta C_i(U)}{\delta y_i}$ at x_i and y_i, respectively are defined as follows.

$$\begin{aligned}
\frac{\delta C_i(U)}{\delta x_i} &= -2bm(x_{i+1} - x_i) + 2bm(x_i - x_{i-1}) \\
&\quad + k\frac{(x_i - p_i)}{\sqrt{(x_i - p_i)^2 + (y_i - q_i)^2}} \\
\frac{\delta C_i(U)}{\delta y_i} &= -2bm(y_{i+1} - y_i) + 2bm(y_i - y_{i-1}) \\
&\quad + k\frac{(y_i - q_i)}{\sqrt{(x_i - p_i)^2 + (y_i - q_i)^2}}
\end{aligned}$$

Setting $\frac{\delta C_i(U)}{\delta x_i} = 0, \frac{\delta C_i(U)}{\delta y_i} = 0$, we get the following two cases. Suppose s_i needs to move left. This means p_i is to the right of the midpoint of nodes s_{i-1} and s_{i+1}. Let $Y_i = \frac{k}{4bm}\frac{1}{\sqrt{1 + \frac{(y_{i-1} + y_{i+1} - 2q_i)^2}{(x_{i-1} + x_{i+1} - 2p_i)^2}}}$. The optimal position is then $x_i = \frac{1}{2}(x_{i-1} + x_{i+1}) + Y_i$. If s_i needs to move right, then p_i is to the left of the midpoint of nodes s_{i-1} and s_{i+1}. The optimal position is then $x_i = \frac{1}{2}(x_{i-1} + x_{i+1}) - Y_i$. The corresponding y_i in both cases is $\frac{(x_{i-1} + x_{i+1} - 2p_i)}{(y_{i-1} + y_{i+1} - 2q_i)}(x_i - p_i) + q_i$.

We note that in some cases it might not be beneficial to move, so the optimal position

for the relay node is its original position. The algorithm to compute the optimal position of a relay node given its neighbors is shown in Fig. 5.7.

function LOCALPOS($o_i, u_i, u_{i-1}, u_{i+1}$)
  ▷ *Consider case s_i moves right*
  $valid \leftarrow FALSE$;
  $x_i \leftarrow \frac{1}{2}(x_{i-1} + x_{i+1}) - Y_i$;
  if $x_i > p_i$ **then**
    $valid \leftarrow TRUE$;
  else
    ▷ *Consider case s_i moves left*
    $x_i \leftarrow \frac{1}{2}(x_{i-1} + x_{i+1}) + Y_i$
    if $x_i < p_i$ **then**
      $valid \leftarrow TRUE$;
    end if
  end if
  ▷ *Record if new position is different from previous one*
  if *valid* **then**
    $y_i \leftarrow \frac{(x_{i-1}+x_{i+1}-2p_i)}{(y_{i-1}+y_{i+1}-2q_i)}(x_i - p_i) + q_i$;

    $u_i' = (x_i, y_i)$;

    if $\|u_i' - u_i\| >$ threshold **then**
      return (u_i', TRUE);
    end if
  end if
  ▷ *not beneficial to move, stay at original position*
  return (o_i, FALSE);
end function

Figure 5.7: Algorithm to compute the optimal position of a relay node that receives data from a single node and transmits the data to a single node.

5.2.3 Static Tree Construction

Different applications may apply different constraints on the routing tree. When only optimizing energy consumption, a shortest path strategy (as discussed below) yields an optimal routing tree given no mobility of nodes. However, in some applications, we do not have the freedom of selecting the routes. Instead, they are predetermined according to some other factors (such as delay, capacity, etc). In other less stringent cases, we may be able to

update the given routes provided we keep the main structure of the tree. Depending on the route constraints dictated by the application, we start our solution at different phases of the algorithm. In the unrestricted case, we start at the first step of constructing the tree. When the given tree must be loosely preserved, we start with the relay insertion step. Finally, with fixed routes, we apply directly our tree optimization algorithm. Our simulations (Section 5.6) show that our approach outperforms existing approaches for all these cases.

We construct the tree for our starting configuration using a shortest path strategy. We first define a weight function w specific to our communication energy model. For each pair of nodes s_i and s_j in the network, we define the weight of edge $s_i s_j$ as: $w(s_i, s_j) = a + b\|o_i - o_j\|^2$ where o_i and o_j are the original positions of nodes s_i and s_j and a and b are the energy parameters discussed in Section 3. We observe that using this weight function, the optimal tree in a static environment coincides with the shortest path tree rooted at the sink. So we apply Dijkstra's shortest path algorithm starting at the sink to all the source nodes to obtain our initial topology.

5.2.4 Node Insertion

We improve the routing tree by greedily adding nodes to the routing tree exploiting the mobility of the inserted nodes. For each node s_{out} that is not in the tree and each tree edge $s_i s_j$, we compute the reduction (or increase) in the total cost along with the optimal position of s_{out} if s_{out} joins the tree such that data is routed from s_i to s_{out} to s_j instead of directly from s_i to s_j using the LocalPos algorithm described in Fig. 5.7. We repeatedly insert the outside node with the highest reduction value modifying the topology to include the selected node at its optimal position, though the node will not actually move until the completion

of the tree optimization phase. After each node insertion occurs, we compute the reduction in total cost and optimal position for each remaining outside node for the two newly added edges (and remove this information for the edge that no longer exists in the tree). At the end of this step, the topology of the routing tree is fixed and its mobile nodes can start the tree optimization phase to relocate to their optimal positions.

5.3 Tree Optimization

In this section, we consider the subproblem of finding the optimal positions of relay nodes for a routing tree given that the topology is fixed. We assume the topology is a directed tree in which the leaves are sources and the root is the sink. We also assume that separate messages cannot be compressed or merged; that is, if two distinct messages of lengths m_1 and m_2 use the same link (s_i, s_j) on the path from a source to a sink, the total number of bits that must traverse link (s_i, s_j) is $m_1 + m_2$.

First, we extend the base case solution of Section 5.2.2 to handle multiple flows passing through a mobile relay node. Then, we propose an iterative algorithm that uses the solution for this base case to compute the new positions of the relay nodes in the routing tree. We also show that this algorithm converges to the optimal solution for the given tree given the topology is fixed.

5.3.1 Extended Base Case

Before we describe our optimal algorithm for this problem, we extend the solution to the base case presented in Section 5.2.2 to the more general multiple flow traffic pattern. The network now consists of multiple sources, one relay node and one sink such that data is

transmitted from each source to the relay node and then to the sink. We modify our solution as follows. Let s_i be the mobile relay node, $S(s_i)$ the set of source nodes transmitting to s_i and s_i^d the sink collecting nodes from s_i. The cost incurred by s_i in this configuration U is:

$$c_i(U) = k\|u_i - o_i\| + am_i + bm_i\|u_d - u_i\|^2$$

where m_i is the total amount of data that s_i transmits to s_i^d. Similar to the single source base case, we define

$$C_i(U) = c_i(U) + \sum_{s_l \in S(s_i)} am_l + b\|u_i - u_l\|^2 m_l$$

This corresponds to the transmission cost of all nodes s_l that send messages to node s_i plus the total cost of node s_i. In this case, this also corresponds to the total cost of configuration U which we wish to minimize. First, we compute m_i as $\sum_{s_l \in S(s_i)} m_l$. We then follow the same routine of computing the points at which both partial derivatives $\frac{\delta C_i(U)}{\delta x_i}$ and $\frac{\delta C_i(U)}{\delta y_i}$ become zero. We obtain the following positions:

$$x_i = p_i + \frac{-B_x(\sqrt{B_x^2 + B_y^2} \pm k)}{A\sqrt{B_x^2 + B_y^2}}$$
$$y_i = q_i + \frac{-B_y(\sqrt{B_x^2 + B_y^2} \pm k)}{A\sqrt{B_x^2 + B_y^2}}$$

where

$$A = m_i + \sum_{s_l \in S(s_i)} m_l$$

$$B_x = m_i x_d + \sum_{s_l \in S(s_i)} m_l x_l + A p_i$$

$$B_y = m_i y_d + \sum_{s_l \in S(s_i)} m_l y_l + A q_i$$

The details of the derivation are provided in proof of Theorem 4 of appendix A. We note that these values correspond to two candidate points moving in each direction (left/right). The optimal position is the valid value yielding the minimum cost.

5.3.2 Optimization Algorithm

We propose a simple iterative approach to compute the optimal position u_i for each node s_i. We define the following notations. Let $u_i^j = (x_i^j, y_i^j)$ be the position of node s_i after the jth iteration of our algorithm for $j \geq 0$ and $U^j = (u_1^j, \ldots, u_n^j)$ the computed configuration of nodes s_1 through s_n after j iterations. We define $u_i^0 = o_i$. Note that the mobile relay nodes do not move until the final positions are computed.

Our algorithm starts by an odd/even labeling step followed by a weighting step. To obtain consistent labels for nodes, we start the labeling process from the root using a breadth first traversal of the tree. The root gets labeled as even. Each of its children gets labeled as odd. Each subsequent child is then given the opposite label of its parent. We define m_i, the weight of a node s_i, to be the sum of message lengths over all paths passing through s_i. This computation starts from the sources or leaves of our routing tree. Initially, we know

$m_i = M_i$ for each source leaf node s_i. For each intermediate node s_i, we compute its weight as the sum of the weights of its children.

```
procedure OptimalPositions(U^0)
    converged ← false;
    j ← 0;
    repeat
        anymove ← false;
        j ← j + 1;
        ▷ Start an even iteration followed by an odd iteration
        for idx = 2 to 3 do
            for i = idx to n by 2 do
                (u_i^j, moved) ← localPos(o_i, S(s_i), s_i^d);
                anymove ← anymove OR moved
            end for
        end for
        converged ← NOT anymove
    until converged
end procedure
```

Figure 5.8: Centralized algorithm to compute the optimal positions in a given tree

Once each node gets a weight and a label, we start our iterative scheme. In odd iterations j, the algorithm computes a position u_i^j for each odd-labeled node s_i that minimizes $C_i(U^j)$ assuming that $u_{i-1}^j = u_{i-1}^{j-1}$ and $u_{i+1}^j = u_{i+1}^{j-1}$; that is, node s_i's even numbered neighboring nodes remain in place in configuration U^j. In even-numbered iterations, the controller does the same for even-labeled nodes. The algorithm behaves this way because the optimization of u_i^j requires a fixed location for the child nodes and the parent of s_i. By alternating between optimizing for odd and even labeled nodes, the algorithm guarantees that the node s_i is always making progress towards the optimal position u_i. Our iterative algorithm is shown in Fig. 5.8

Fig. 5.9 shows an example of an optimal configuration for a simple tree with one source node. Nodes start at configuration U^0. In the first iteration, odd nodes (s_3 and s_5) moved to their new positions (u_3^1, u_5^1) computed based on the current location of their (even) neighbors

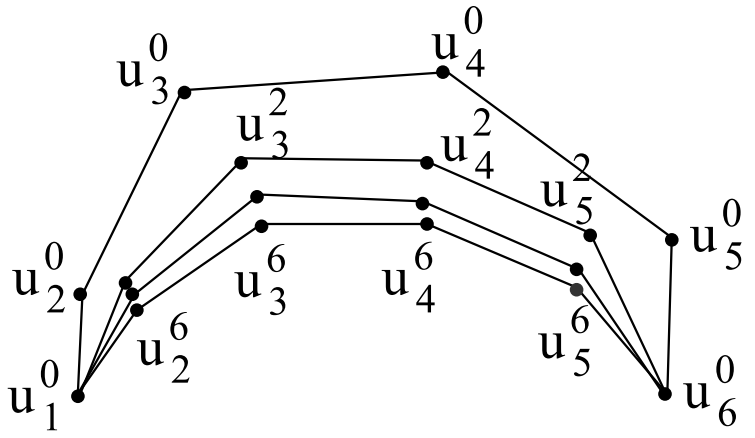

Figure 5.9: Convergence of iterative approach to the optimal solution. Each line shows the configuration obtained after 2 iterations. The optimal configuration is reached after 6 iterations.

(u_2^0, u_4^0, u_6^0). In the second iteration, only even nodes (s_2 and s_4) moved to their new positions (u_2^2, u_4^2) computed based on the current location of their (odd) neighbors (u_1^1, u_3^1, u_5^1). Since s_3 and s_5 did not move, their position at the end of this iteration remains the same, so $u_3^1 = u_3^2$ and $u_5^1 = u_5^2$. In this example, nodes did two more sets of iterations, and finally converged to the optimal solution shown by configuration U^6.

Even though configurations change with every iteration, nodes only move after the final positions have been computed. So each node follows a straight line to its final destination. As the data size increases, nodes in the optimal configuration get more evenly spaced. In fact, in any given configuration, the maximum distance traveled by a node is bounded by the distance between its starting position and its final position in the evenly spaced configuration.

The above example shows another property of our algorithm. When a node s_i moves and its neighbors (s_{i-1} and s_{i+1}) remain in place, it moves in the direction of the midpoint of $s_{i-1}s_{i+1}$. This results in a reduction in the length of one of the transmission links. The other may increase in length but will never exceed the new length of the first link. This

remains valid for multiple children case. So in any configuration U^{i+1}, the length of the largest link is at most the length of the largest link in the previous configuration U^i. So if we start with a route along good quality links, this quality will be preserved in the optimal configuration (and throughout intermediate configurations).

5.4 Efficiency and Optimality

We first consider efficiency. Our initial tree construction algorithm is essentially a single source shortest path algorithm. Using Dijkstra's algorithm, the time complexity is $O(n^2)$ where n is the number of nodes. Our second algorithm needs to compute the reduction in cost for each pair of node and tree edge, so the time complexity is $O(n^2)$. Our tree optimization algorithm runs until the change in position for each node falls below a predefined threshold. The value of this threshold represents a tradeoff between precision and cost. As the threshold decreases, more iterations are needed for convergence. Upon termination, no node can move by itself to improve the overall cost (within the threshold bound). We have not completed a rate of convergence analysis for this algorithm. However, in our simulations, we reach our error threshold within 8 to 10 iterations. Since each iteration involves only half the nodes and each computation of u_i^j can be performed in constant time, the time complexity of our algorithm is $O(\lambda n)$, where λ is the number of iterations to reach convergence. Given that $\lambda \leq 10$ in our simulations, our observed time complexity is $O(n)$. The resulting time complexity for the full approach is $O(n^2)$.

With respect to optimality, our resulting configuration is not necessarily optimal because we do not necessarily find the optimal topology. However, two of our algorithms, the initial tree construction algorithm and the tree optimization algorithm, are optimal for their re-

spective subproblems. That is, our initial tree construction algorithm is optimal in a static environment where nodes cannot move so that only the original positions of the nodes are considered. Likewise, for our tree optimization algorithm, we prove that the final configuration where no node can move by itself to improve the overall cost (within the threshold bound) is globally optimal; that is, no simultaneous relocation of multiple nodes can improve the overall cost. We present the proof of optimality in Theorem 5 of appendix A. The key intuition is that for a configuration in which no relay node can move and improve the cost by itself, the directional derivative [7] in any direction at that configuration is positive; this is a sufficient condition for the optimality of that configuration because the cost function is convex.

5.5 Distributed Algorithms

Our solutions to the three subproblems assume a centralized scheme in which one node has full knowledge of the network including which nodes are on the transmission paths to each source, the original physical position o_i of each node s_i, and the total message length m to be sent from each source. Whereas the centralized algorithm computes the optimal static tree and the optimal position of each node in the restructured tree, it incurs prohibitively high overhead in large-scale networks. We now present a distributed and decentralized version of each of our algorithms.

We modify the first phase, the tree construction phase, to use a fully distributed routing algorithm. We pick greedy geographic routing since it does not require global knowledge of the network although any algorithm with such property can be used.

After a routing tree is constructed, the tree restructuring phase begins. Network nodes

outside the tree broadcast their availability (as NODE_IN_RANGE message) to tree nodes within their communication range and wait for responses for a period of time T_w. Similarly, tree nodes enter a listening phase T_u. During that period, tree nodes receive messages of different types (NODE_IN_RANGE, OFFER, ...). Each tree node that receives one or more NODE_IN_RANGE message responds to the sender by giving it its location information and its parent's location information. Each non-tree node s_o that receives location information from a tree node s_i during T_w computes the reduction in cost if it joins the tree as parent of s_i and adds s_i to a list of candidates. At the end of T_w, the non-tree node selects from the candidate list the node that results in the largest reduction and sends it an offer. It also sends the tree node with the second largest reduction a POTENTIAL_OFFER message. At the end of T_u, each tree node v_t that collected one or more offers and potential offers operates as follows. If v_t's best potential offer exceeds its best offer by a certain threshold B and v_t has not already waited R rounds, v_t waits rather than accepting its best offer in the hopes that its best potential offer will become an actual offer in another round. By waiting, it sends everyone a REJECT_OFFER, restarts the listening phase, and records that it has waited another round. Otherwise, v_t accepts its best offer by responding to its sender p with an ACCEPT_OFFER message and to the remaining nodes with a REJECT_OFFER message. It then updates its parent in the tree to p, resets T_u and starts the listening phase again.

A non-tree node p that receives an ACCEPT_OFFER message moves to the corresponding local optimal location and joins the tree. It becomes a tree node and enters the listening phase. On the other hand, if p does not receive an ACCEPT_OFFER, p repeats the process by broadcasting its availability again and resetting T_w. We note that values in p's candidate

```
procedure treeRun
    ▷ Phase I: Run routing algorithm to discover parent and children
    (parent, children) ← DistributedRouting;
    ▷ Phase II: Start tree restructuring phase
    offers ← ∅; potentialoffers ← ∅; wait ← 0;
    repeat
        ▷ Listen to incoming offers or changes in structure
        repeat
            receive(sender, type, data);
            if type = MOBILE_IN_RANGE then
                send(sender, META_DATA, info);
            else if type = OFFER then
                offers.add(data);
            else if type = POTENTIAL_OFFER then
                potentialloffers.add(data);
            else if type = UPDATE_STRUCTURE then
                children.add(data.newchild);
                children.remove(data.oldchild);
            end if
        until timeout
        ▷ Process offers and pick best
        if offers ≠ ∅ then
            bestOffer ← offers.dequeue();
            bestPotentialOffer ← potentialoffers.dequeue();
            if bestPotentialOffer > bestOffer*B and wait < R then
                send(bestOffer.sender, REJECT_OFFER);
                wait++;
            else
                send(bestOffer.sender, ACCEPT_OFFER);
                parent ← bestOffer.sender;
            end if
        end if
        while offers ≠ ∅ do
            offer ← candidates.dequeue();
            send(offer.sender, REJECT_OFFER);
        end while
    until timeout
    ▷ Phase III: Iterate moving to optimal local positions
    converged ← false;
    while not converged do
        (u, converged) ← LocalPos(o, parent, children);
        ▷ Exchange location info with parent and children
        send(parent, NEW_LOCATION, u);
        for all child ∈ children do
            send(child, NEW_LOCATION, u);
        end for
        receive(parent, NEW_LOCATION, parent.u);
        for all child ∈ children do
            receive(child, NEW_LOCATION, child.u);
        end for
    end while
end procedure
```

Figure 5.10: Local algorithm executed by tree nodes

list cannot be reused to extend offers to old tree nodes since those tree nodes could have a new parent at this point in time. When the second phase ends, any remaining non-tree nodes stop processing whereas tree nodes enter the tree optimization phase. Fig. 5.10 shows the algorithm executed by each tree node.

Giving tree nodes the ability to wait before accepting an offer increases the chances of using mobile relay nodes to their full potential. For example, consider a scenario where several mobile relay nodes can greatly improve the capacities of several tree links but are all closest to one specific link. They will all send offers to the same tree node while the rest of the tree nodes in their proximity will receive modest offers from more distant mobile nodes. If the tree nodes cannot wait, they will be forced to accept a modest offer and the mobile nodes will either remain unused or they will help more distant tree nodes where their impact is reduced since they use up more energy to get to their new location.

The centralized tree optimization algorithm can be transformed into a distributed algorithm in a natural way. The key observation is that computing each u_i^j for node s_i only depends on the current position of s_i's neighbors in the tree (children and parent), nodes that s_i normally communicates with for data transfers. Thus, s_i can perform this computation. The distributed implementation proceeds as follows. First, there is a setup process where the sender s_1 sends a discover message that ends with the receiver s_n; the two purposes of this message are (1) to assign a label of odd or even to each node s_i and (2) for each node s_i to learn the current positions of its neighbors. A node s_i sends its current position to node s_j when acknowledging receipt of the discover message. Second, there is a distributed process by which the nodes compute their transmission positions. We make each iteration of the basic algorithm a "round", though there does not need to be explicit synchronization.

In odd rounds, each odd node computes its locally optimal position and transmits this new position to its neighbors. In even rounds, each even node does the same. A node begins its next round when it receives updated positions from all its neighbors. The final step is to have the nodes move to their computed transmission positions, send messages to their neighbors saying they are in position, and finally perform the transmission. To ensure the second process does not take too long, we limit the number of rounds to 8; that is, each node computes an updated position four times. Simulation results show that this is enough to obtain costs close to optimal (see Section 5.6).

5.6 Simulations

5.6.1 Setup

We carried out simulations on 100 randomly generated initial topologies, each of which has 100 nodes placed uniformly at random within a 150m by 150m area. We used these initial topologies to generate two subsequent sets of complete topologies with established sources and sink. We used the first set to study the effectiveness of our algorithms as the amount of data transferred to the sink varies and the second set to study the effectiveness of our algorithms for different numbers of sources. In the first set, we selected sources and sinks uniformly at random from these 100 nodes. We varied the number of sources from 4 to 12, by increments of 2, and used each number of sources for 20 initial topologies. For each resulting topology, we created many separate input instances by varying the data chunk size from 1MB to 150MB where the data chunk size for an input instance is the common amount of data to be transferred from each source to the sink. In the second set, for each initial

topology, we generated 10 different complete topologies by starting with 2 randomly selected sources, and adding two new sources to the previous set at each step.

We used the following settings to model the transmission and mobility costs of our nodes. For transmission, we use $a = 0.6 \times 10^{-7}$ and $b = 4 \times 10^{-10}$ as the standard setting which is consistent with the empirical measurements on CC2420 motes [45]. For mobility, we used different settings in each of our two sets. In the first set, we used $k = 2$ as the standard setting because it models several platforms such as Robomote [12, 1]. In the second set, we set k to be 1, 2 and 4 since we additionally use that set to study the effect of different mobility costs on the energy reduction. Furthermore, we set the maximum communication distance of a node to be 30m, which was shown to result in a high packet reception ratio for the CC2420 radio [45]. We ran simulations using different values for the convergence threshold. We obtained similar gains for values less than or equal to 0.01. In the following simulations, we set the threshold to 0.01.

Our algorithmic framework starts with an initial routing tree. In the centralized setting, we construct this initial routing tree using the following three widely used routing algorithms: power based routing, hop based routing, and greedy geographic routing. Power based routing computes a shortest path from the sink to each source with each edge weight being the square of the distance between the two corresponding nodes plus some constant value to represent the energy consumed $a + bd^2$ to transmit each byte of data over that edge. Hop based routing minimizes the number of hops between each source and the sink and is the base of several widely used algorithms in wireless networks (e.g. AODV [40]). Given our maximum communication range of 30m, we do not have any links with poor quality which is a common concern with hop based routing. Greedy geographic routing is a greedy strategy in which

each node forwards messages to the reachable node (within the communication range of the node) that is closest to the sink. The first two tree construction approaches require global knowledge of the network whereas the last one is fully localized. For the distributed setting, we construct the initial routing tree using greedy geographic routing because it is fully localized. Of the 100 initial topologies, the distributed routing algorithm resulted in a disconnected path between the sources and the sink in only four networks given our maximum communication distance of 30m.

5.6.2 Performance Metrics

We study variants of our strategy where we use only one optimization, inserting nodes or optimizing a given tree, to determine the benefit of both optimizations. Specifically, we use TREE to represent the variant where we only construct an initial tree and do no optimizations, TREE+FO to represent the variant where we optimize the initial tree, TREE+INS to represent the variant where we insert nodes into the initial tree, and TREE+INS+FO to represent the variant where we insert nodes into the initial tree and then optimize the final tree. The three possibilities for TREE are PB, HB, and GG which represent the Power Based, Hop Based, and Greedy Geographic tree construction algorithms, respectively. For each input instance I, we let $TREE(I)$ denote the energy consumed by the initial tree constructed by our three tree construction algorithms PB, HB, and GG, and we let $TREE+OPT(I)$ denote the energy consumed by the final optimized tree where TREE can be PB, HB, or GG and OPT can be INS, FO or INS+FO. The *reduction ratio* achieved by optimization OPT on input I for tree construction algorithm TREE is $(TREE(I) - TREE + OPT(I))/TREE(I)$. We measure the performance of optimization OPT on initial tree strategy TREE by computing

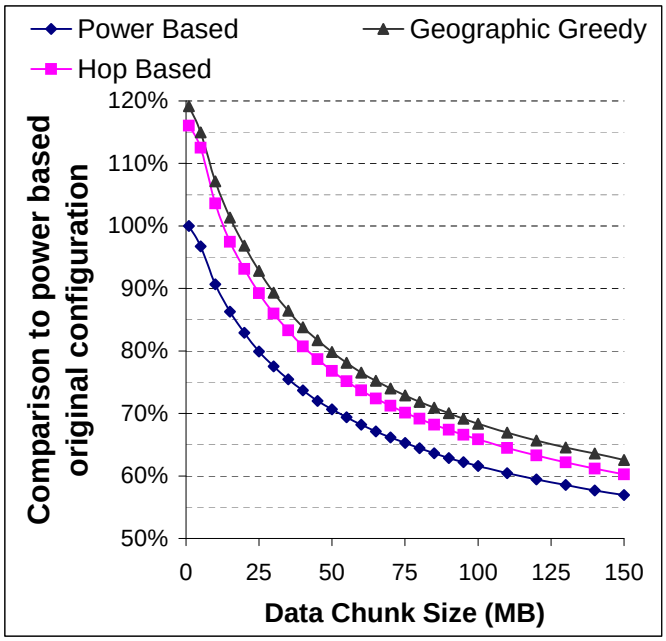

Figure 5.11: Graph of the average static energy consumption ratio of TREE+INS+FO as a function of data chunk size for our three tree construction strategies PB, HB, and GG

the average reduction ratio achieved by OPT over all input instances I of set 1 that have the same data chunk size. Moreover, for each input instance I and each algorithm TREE+OPT, we define the *static energy ratio* $(TREE + OPT(I))/PB(I)$ where $PB(I)$ is the cost of the power based tree which is the optimal cost for the static version of this problem where no nodes can move. The static energy ratio measures the benefit of our algorithms which exploit mobility of nodes versus the static optimal configuration. We measure the overall performance of algorithm TREE+OPT by computing the average static energy ratio achieved by TREE+OPT over all input instances I of set 1 that have the same data chunk size. Finally, we measure the performance of optimization INS+FO on initial tree strategy TREE by computing the average reduction ratio achieved by INS+FO over all input instances I of set 2 that have the same number of sources.

5.6.3 Centralized Algorithm

We first show the benefit of exploiting the mobility of relay nodes by computing the average static energy consumption ratio of TREE+INS+FO for all data chunk sizes for each of our three tree building strategies PB, HB, and GG as shown in Fig. 5.11. For all three initial tree strategies, we see that the average static energy consumption ratio drops quickly as the data chunk size increases. For HB and GG, the average static energy consumption ratio starts out higher than 100% because $PB(I)$, the optimal tree for the static case, is roughly 37% lower than $HB(I)$ and $GG(I)$ for any of our input instances. Even given this initial disadvantage of a poor starting tree from an energy consumption perspective, we see that the average static energy consumption ratios of HB+INS+FO and GG+INS+FO drop below 100% for data chunk sizes of 12 MB and 15 MB, respectively. As the data chunk size increases further, both HB+INS+FO and GG+INS+FO achieve average static energy consumption ratios of 75% and 60% for data chunk sizes of 60 MB and 150MB, respectively. The results for PB+INS+FO are even better because we start with the optimal tree for the static case. Thus, the average static energy consumption ratio for PB+INS+FO is always below 100% and reaches 55% for 150 MB.

We now evaluate the benefit achieved by our optimizations FO, INS, and INS+FO for each of our tree building strategies PB, HB, and GG. We note that in this set of simulations, we used our centralized improvement schemes with the distributed tree building approach GG. The purpose is to test the limits of our optimizations given a non-optimal starting tree. A fully distributed setup is studied later in this section.

We start with optimization INS+FO. Fig. 5.12 plots the average reduction ratio for

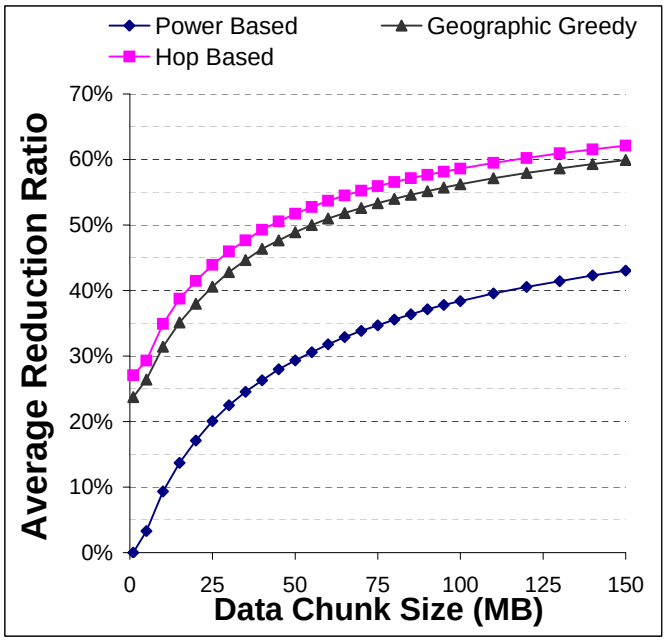

Figure 5.12: Graph of the average reduction ratio of optimization INS+FO as a function of data chunk size for our three tree construction strategies PB, HB, and GG

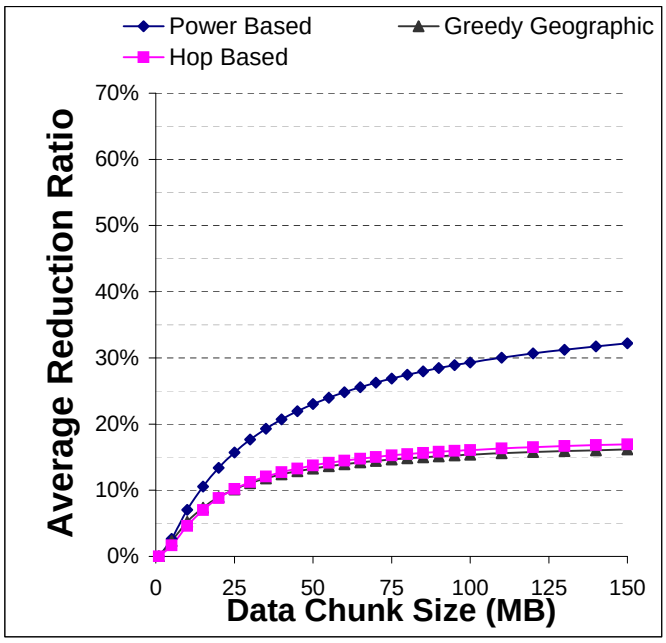

Figure 5.13: Graph of the average reduction ratio of optimization FO as a function of data chunk size for our three tree construction strategies PB, HB, and GG

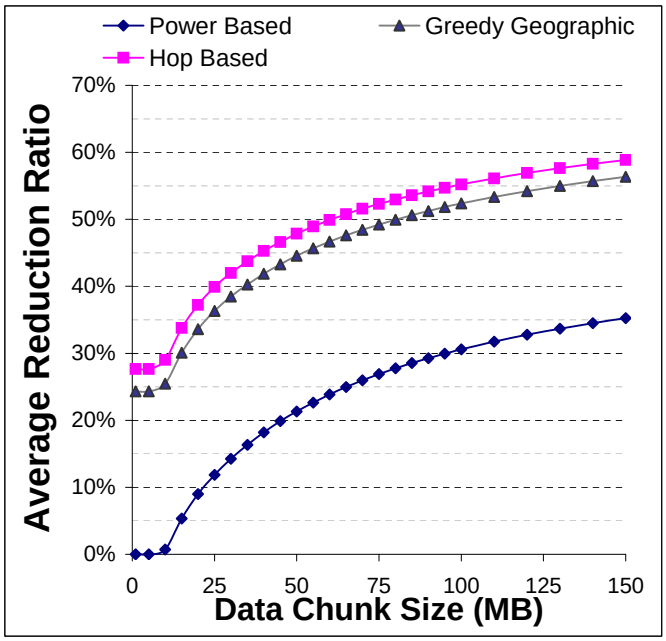

Figure 5.14: Graph of the average reduction ratio of optimization INS as a function of data chunk size for our three tree construction strategies PB, HB, and GG

optimization INS+FO for PB, HB, and GG. In all three cases, we see the same basic trend; the average reduction ratio increases as data chunk size increases. For both HB and GG, the average reduction ratio starts at roughly 25% for small data chunk sizes and exceeds 60% for large data chunk sizes; for PB the average reduction ratio starts near 0% and exceeds 43% for large data chunk sizes. The difference in average reduction ratio, in particular for small data chunk sizes, is due to the quality of the initial tree. For PB, the initial tree is good so there is little that our optimization INS+FO can do to improve energy consumption for small data chunk sizes. For HB and GG, the initial tree can be very poor, so INS+FO can provide immediate improvement to the tree to significantly reduce energy consumption by an average of 25% for data chunk sizes of only 1MB. We note that although INS+FO achieves higher reduction ratios for HB and GG than for PB, the total energy consumed by PB+INS+FO is lower than the total energy consumed by HB+INS+FO or GG+INS+FO.

We next consider optimization FO alone. Fig. 5.13 plots the average reduction ratio for optimization FO for PB, HB, and GG. In all three cases, the average reduction ratio starts at 0% for small data chunk sizes and increases to roughly 18% for HB and GG and 33% for PB for large data chunk sizes. It is interesting to note that FO is most effective for PB whereas INS+FO achieved significantly greater reduction ratios for GG and HB for all data chunk sizes.

Finally, we consider optimization INS alone. Fig. 5.14 plots the average reduction ratio for optimization INS for PB, HB, and GG. In all three cases, we see the average reduction ratio of INS alone is comparable to that of INS+FO (within 5%-8% for data chunk sizes of at least 15MB). For very small data chunk sizes, the average reduction ratio is constant until a certain threshold is exceeded and then rises significantly.

We now evaluate our approach as we vary the number of sources. We used the greedy geographic tree GG as our initial tree and INS+FO as our optimization algorithm. Fig. 5.15 shows the average reduction ratio as a function of the number of sources. We observe that this ratio remains almost constant for different values of k as the difference in ratios for different number of sources does not exceed 3.5%. Fig. 5.15 also shows the effect of mobility costs on the reduction in energy consumption costs in general. As mobility costs decrease, it becomes more effective for mobile nodes to move over longer distances and reduce the communication consumption further so the reduction in total costs increases as k decreases.

Given our simulation results, we draw the following five conclusions. First, we achieve the best results when we use the power based tree PB as our initial tree. Second, if we use the power based tree, either optimization alone is very effective and both optimizations together achieve the best results. Third, if we start with either the hop based tree HB or the greedy

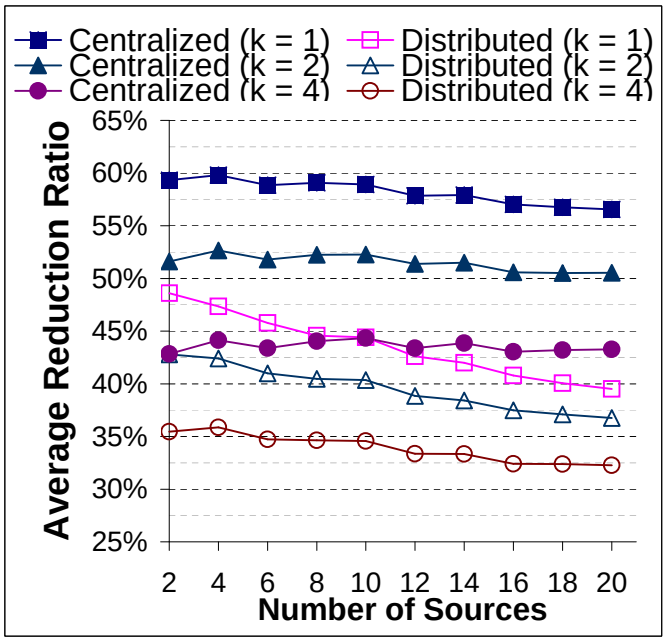

Figure 5.15: Graph of the average reduction ratio of the centralized and distributed GG+INS+FO optimizations as a function of the number of sources, a data chunk of 75MB and different values of k.

geographic tree GG, the most effective optimization is the node insertion optimization INS which achieves nearly as good an average reduction ratio as INS+FO. Fourth, if we start with the hop based or greedy geographic tree, we can achieve a static energy ratio that is close to that achieved by starting with the power based tree if we apply both optimizations. In particular, the node insertion optimization INS helps alleviate the initial disadvantage by adding a lot of new nodes into the tree. We briefly explain the reason for all of these conclusions. The key observation is that the hop based and greedy geographic trees HB and GG tend to create initial trees with relatively long edges and relatively few nodes whereas the power based tree PB tends to create trees with lots of nodes and relatively short edges because of the quadratic cost metric. As a result, for HB and GG, optimization FO alone which rearranges nodes is relatively ineffective as it can only balance the relatively long edges. On the other hand, optimization INS alone can insert new nodes into the tree and

thus create a new tree with significantly shorter edges on average given HB or GG as the initial tree. Because PB starts with many more nodes and shorter edges, PB does not benefit as much from node insertion INS as HB and GG do, and PB benefits a lot more from node rearrangement FO than HB and GG do. Fifth, the improvement ratios that we obtain are almost independent of the number of sources in the network.

In all our simulation results, the standard deviation varied between 4% and 6.5%. We identified six outlier topologies which deviated from the mean by more than 10%. In these topologies, the sources were either very close to the sink so there was little room for improvement or very far from the sink so the improvement was much greater than the average case.

5.6.4 Distributed Algorithm

We now evaluate how well our distributed implementation works. Our initial tree is the greedy geographic tree GG. We consider four optimizations: the centralized implementation of INS+FO, the distributed implementation of just FO, the distributed implementation of just INS, and the distributed implementation of INS followed by the distributed implementation of FO. For the distributed implementation of INS, we set parameter B to 10% (a potential offer must be 10% better than the best actual offer to cause a node to wait). Fig. 5.16 shows the average reduction ratio of each of these optimizations.

The average reduction ratio for distributed INS+FO starts at 20% for small data chunk sizes, reaches 30% for data chunk sizes around 20MB, and exceeds 40% for data chunk sizes larger than 75MB. The gap between the average reduction ratio for centralized INS+FO and distributed INS+FO starts at roughly 5% for small data chunk sizes and increases to

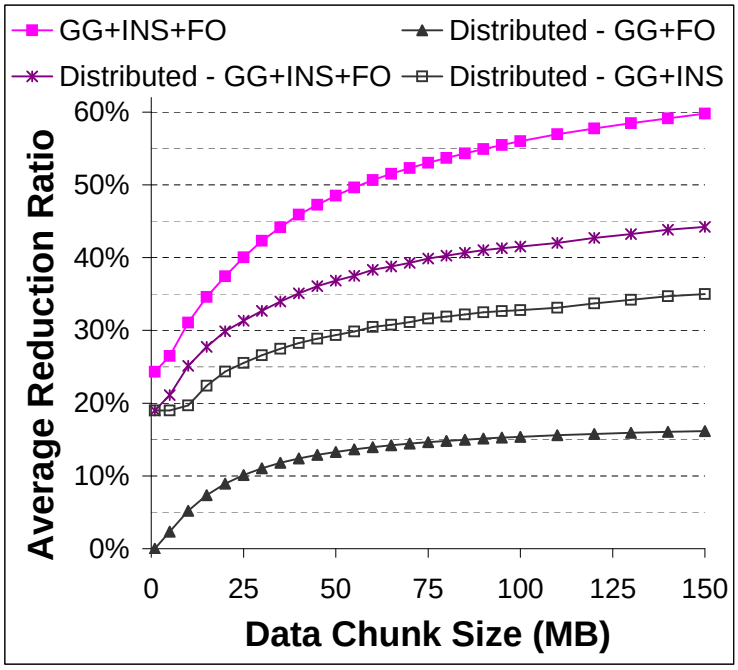

Figure 5.16: Graph of the average reduction ratio of the centralized optimization (INS+FO) and three distributed optimizations (FO, INS, INS+FO) as a function of data chunk size for the greedy geographic tree GG.

roughly 15% for large data chunk sizes. This gap is due to the lack of global information when performing the insertion step. Expensive links in the tree that do not have nearby relay nodes are not able to communicate with further but available relay nodes whose help is only offered to cheaper but nearby links. This problem is exacerbated as the data chunk size increases. We varied values for B between 10% and 50% and for R between 1 and 3. For all combinations of B and R that we tested, we obtained similar results to those of Fig. 5.16. As in the centralized case, distributed INS is more effective than distributed FO. However, doing both distributed optimizations does result in roughly a 10% improvement compared to only doing the distributed INS optimization for most data chunk sizes.

Similar to the centralized implementation, we observe a slow reduction in the improvement ratio as the number of sources increases for $k = 2$ and 4 (Fig. 5.15). For cheaper

mobility cost ($k = 1$), the difference in improvement ratios increases at a faster rate and reaches 9% as the number of sources increases from 2 to 20. This is because when mobility is cheaper, in an optimal setting, nodes can move over longer distances to help expensive links. However, as we mentioned earlier, in a distributed setting, mobile nodes are not aware of those distant expensive edges. Moreover, as the number of sources increases, the number of mobile nodes available to help decreases. Both factors combined make the distributed implementation slightly less effective for a high number of sources.

5.7 Summary

In this chapter, we proposed a holistic approach to minimize the total energy consumed by both mobility of relays and wireless transmissions. When we model both sources of energy consumption, the optimal position of a node that receives data from one or multiple neighbors and transmits it to a single parent is not the midpoint of its neighbors; instead, it converges to this position as the amount of data transmitted goes to infinity. Ideally, we start with the optimal initial routing tree in a static environment where no nodes can move. However, our approach can work with less optimal initial configurations including one generated using only local information such as greedy geographic routing. Our approach improves the initial configuration using two iterative schemes. The first inserts new nodes into the tree. The second computes the optimal positions of relay nodes in the tree given a fixed topology. This algorithm is appropriate for a variety of data-intensive wireless sensor networks. It allows some nodes to move while others do not because any local improvement for a given mobile relay is a global improvement. This allows us to potentially extend our approach to handle additional constraints on individual nodes such as low energy levels or

mobility restrictions due to application requirements.

CHAPTER 6

Maximizing Network Lifetime Using Node Rotation

In this chapter, we directly address the problem of extending the network lifetime. Similar to the previous setting, we consider networks that consist entirely of mobile nodes. Additionally, all the nodes in the network generate data that need to be relayed to the sink. Most sensor networks have a many-to-one traffic pattern and consequently a tree topology. One of the main reasons limiting the lifetime of such networks is that nodes that are closer to the sink usually transmit a large amount of data and consequently drain their battery at a much faster rate than nodes further from the sink. As a result, the lifetime of the network is substantially reduced even though the network still contains nodes rich in energy. We propose an approach that exploits the mobility of the nodes to balance the energy consumption throughout the network. The main idea is to redistribute the nodes in the network such that nodes at low traffic (i.e. low consumption) positions switch places with nodes at high traffic (i.e. high consumption) positions.

The rest of this chapter is organized as follows. We first describe the motivation behind our approach in Section 6.1. Then, we define the problem of maximizing the network lifetime using node rotations in Section 6.2. In Section 6.3, we present our solutions for the problem. In Section 6.4, we derive some upper bounds on the effectiveness of any node rotation algorithm for several classes of input instances. Section 6.5 describes our simulation results and Section 6.6 summarizes the chapter.

6.1 Motivation

Recently, the *controlled mobility* of sensors has been exploited to improve the energy efficiency of WSNs. For instance, by relocating mobile sensors, the communication topology of a network can be dynamically configured to reduce power consumption. Moreover, mobile sensors can physically carry large chunks of data to reduce energy consumption in wireless transmissions [50]. Such approaches become increasingly attractive due to the emergence of numerous low-cost mobile sensor prototypes such as Robomote [12], Khepera [1], and FIRA [28].

However, many applications have constraints which make existing approaches infeasible. We identify three key constraints. The first is that the location of the nodes and the communication topology of the network may not be mutable. For example, in a surveillance or environment monitoring application, the exact placement of sensor nodes may not be adjusted without compromising the operation of the system. The second is that all nodes have equal, typically limited, capabilities. This rules out approaches that require a few nodes with extra capabilities and the ability to perform complex motion planning. The third is that nodes face differential power consumption where some nodes consume significantly more

power than other nodes. For example, nodes closer to the sink in a given routing topology often have to transmit more data and thus consume more power than nodes farther from the sink in the given topology.

To address these three constraints that limit existing techniques, we propose a new approach that we call *mobile node rotation* which is inspired by the huddling and rotation behavior of emperor penguins that help them breed in the fierce arctic winter. Penguins on the outside of the huddle face temperatures as low as $-45\,^{\circ}\mathrm{C}$ and strong winds while those on the inside of the huddle enjoy warm ambient temperatures as high as $37\,^{\circ}\mathrm{C}$ and significant wind protection. Emperor penguins rotate positions to share the burden of being on the outside [69]. In mobile node rotation, we propose to rotate the physical positions of mobile sensors to share the burden of any high power consumption location. We observe that mobile node rotation is particularly suitable for mobile sensor platforms with limited mobility as we can impose mobility constraints on individual nodes. For example, we can model the constraints of the NIMS sensors [41] that are only capable of moving along fixed cables by only allowing such sensors to exchange with other sensors on the same set of cables. Likewise, mobile node rotation does not require powerful nodes capable of performing complex motion planning calculations or developing new mobility-aware routing topologies since all movements are to known positions and the topology does not change.

We first present a new problem, *Max-lifetime Node Rotation (MaxLife)*, that models maximizing the lifetime of a WSN using rounds of mobile node rotation. MaxLife can incorporate any energy consumption model for both wireless communication and node movement. We then efficiently solve the one round MaxLife problem by reducing it to the assignment problem and the general multiple round MaxLife problem by performing localized rotations

of node pairs. We prove upper bounds on the lifetime improvement ratio of mobile node rotation approaches. Finally, we conduct extensive simulations based on energy models obtained from existing mobile sensor platforms. We show that our algorithms can significantly increase the network lifetime. With just one rotation round, we can almost double the network lifetime. With multiple rotation rounds, we can increase network lifetime by more than a factor of seven.

6.2 Problem Definition

We consider WSNs consisting of many wireless mobile sensor nodes and a single static sink. The sensor nodes gather data from their surroundings and transmit the data through one or multiple hops to the sink forming a directed routing tree. We divide time into intervals. In each interval, each sensor node transmits the data it gathered as well as the data it received from its children to its parent along the routing tree. The goal is to maximize the lifetime of the WSN, i.e. the number of time intervals until the first node dies. We use this definition of lifetime assuming that all nodes are needed in their exact positions in order to not compromise the operation of the system.

One of the main reasons limiting the lifetime of such networks is differential power consumption; nodes closer to the sink usually transmit a large amount of data and consequently consume more power than nodes further away from the sink. As a result, the lifetime of the WSN is substantially reduced even though the WSN still contains nodes rich in energy. Mobile base stations mitigate differential power consumption by having a powerful mobile sink move around the WSN. We propose a new solution, mobile node rotation, that uses multiple low-cost mobile nodes that rotate or swap positions and roles allowing nodes to share the

burden of high consumption locations and the benefits of low consumption locations. We formally define the problem as follows.

Definition 2 (One-Round Max-Lifetime Node Rotation (1-MaxLife))**.**

Input Instance:

- $S = (s_1, \ldots, s_n)$*, a list of sensor nodes*
- u*, the network sink, and* p_u*, its position*
- $P = (p_1, \ldots, p_n)$*, a list of positions such that node* s_i *starts at position* p_i
- $E = (e_1, \ldots, e_n)$*, a list of initial energies for nodes in* S
- T*, a directed routing tree represented as a set of non-zero values* t_{ij} *for every arc* (p_i, p_j) *in the tree corresponding to the amount of energy consumed when transmitting one data unit from* p_i *to* p_j
- K*, a set of values* k_{ij} *for every pair of positions* p_i *and* p_j*, corresponding to the amount of energy consumed by a node when it moves between* p_i *and* p_j
- $\Lambda = (\lambda_1, \ldots, \lambda_n)$*, the amount of data gathered at each position per time interval*

Output Instance: A matching M *of nodes in* S *to locations in* P*, and two durations* r_1 *and* r_2 *such that nodes transmit data from their original positions for* r_1 *time intervals, relocate to their new position according to* M*, then generate and transmit data for* r_2 *time intervals such that the total duration* $(r_1 + r_2)$ *is maximized and no node's energy goes to 0 before* $r_1 + r_2$*.*

We define the **Max-Lifetime Node Rotation (MaxLife)** problem to be the general version where nodes can switch positions any number of times.

6.3 Node Rotation Algorithms

In this section, we first present our centralized node rotation algorithm NR1 to the 1-MaxLife problem. For comparison purposes, we present CNR, an extended version of NR1 that applies to the general MaxLife problem. Finally, we present a practical distributed algorithm DNR for the general MaxLife problem that uses only local information and reduces the number of node movements. All algorithms begin by having nodes compute the load l_j at each position p_j in P which is the total energy consumed in transmitting all the data gathered in one time interval from the subtree rooted at p_j to its parent. More formally, $l_j = t_{jq} \sum_{i \in T(p_j)} \lambda_i$ where $T(p_j)$ is the subtree rooted at p_j and p_q is the position of the parent of p_j in the tree.

6.3.1 Algorithm NR1

NR1 transforms the input instance into an instance of the assignment problem [8] (outlined in Section 4.2.2). We first assume that we know the optimal length of the first time interval r_1 for which nodes remain at their initial positions. To compute the optimal matching M of sensor nodes to positions, we transform the instance I into an instance of the maximum bottleneck assignment problem I' for the given r_1 as follows. Each mobile node s_i corresponds to a person and each location p_j in P corresponds to a task. The efficiency c_{ij} of a person s_i performing task p_j corresponds to the total lifetime of s_i after transmitting for a period r_1 from its original position p_i, then moving to p_j where it transmits until its energy is depleted; that is, $c_{ij} = r_1 + \frac{e - l_i r_1 - k_{ij}}{l_j}$. The optimal solution for the maximum bottleneck assignment instance I' corresponds to an optimal matching M for the given r_1.

We now need to compute the best duration r_1 such that the total duration of trans-

```
procedure NR1(S, P, F, E)
    ▷ Initialize empty priority queue
    GOLDEN_RATIO ← 2/(3+√5);
    rlo ← 0;
    rhi ← StaticLifetime(S, P, F, E);

    ▷ Run golden ratio search for best r
    while rhi - rlo ≥ error do
        ra ← rlo + GOLDEN_RATIO*(rhi - rlo);
        rb ← rlo - ra + rhi;
        (La, Π) ← optimalAssignmentValue(S, P, E, F, ra);
        (Lb, Π) ← optimalAssignmentValue(S, P, E, F, rb);
        if Lb ≤ La then
            rhi ← rb;
        else
            rlo ← ra;
        end if
    end while
return (ra, La, Π);
end procedure


procedure optimalAssignmentValue(S, P, E, F, r)
    ▷ Build efficiency matrix
    for all s_i ∈ S do
        for all p_j ∈ P do
            if s_i.location = p_j then
                c_ij ← e_i/l_j
            else
                if e_i − l_i r − k d_ij > 0 then
                    c_ij ← (1 − l_i/l_j) r + (e_i − k d_ij)/l_j
                else
                    c_ij ← 0
                end if
            end if
        end for
    end for
    ▷ Find optimal matching
    (L, Π) ← SolveBottleneckAssignment(c)
return (L, Π)
end procedure
```

Figure 6.1: Algorithm NR1 for computing the optimal lifetime through a single round of rotations

missions ($r1 + r2$) is maximized. We observe that as r_1 increases, r_2 decreases since there is less energy for the second round. So we can express r_2 as a function $L_2(r_1)$. We can then define the total network lifetime as a function $L(r_1) = r_1 + L_2(r_1)$. We use golden ratio search to find the best r_1. When $L(r_1)$ is unimodal, the golden ratio search yields an optimal r_1 that maximizes $L(r_1)$. To start the golden ratio search algorithm, we first compute $L(I) = \min_{j=1}^{n} e_j/l_j$ which is an upper bound on r_1. Our algorithm NR1 runs in $O(\log L(I) n^{2.5})$ time because the golden ratio search has $O(\log L(I))$ time complexity and each assignment problem has $O(n^{2.5})$ time complexity. Algorithm NR1 is shown in Fig. 6.1.

6.3.2 Centralized Algorithm CNR(r)

CNR(r) extends NR1 as follows. First, a node is selected to be the controller. The controller collects energy and location information from all the other nodes once. CNR(r) then proceeds in rounds, each of which has a common duration r. At the beginning of each round, the controller computes matchings of nodes to positions for the following round using duration r as each round length until the first node dies. We note that the controller has enough information to estimate each node's available energy for each future round. The controller then broadcasts the number of rounds and all the matchings to the other nodes. Finally, the nodes carry out the rounds synchronizing as needed to initiate each rotation. We present CNR(r) solely for benchmarking purposes. We show CNR in Fig 6.2.

6.3.3 Distributed Algorithm DNR

Our practical multiple rotation round algorithm is DNR(r, h, l_{cr}, f) which requires only local information. We typically drop the four parameters and refer to this algorithm as DNR. The main idea is that only critical nodes, *i.e.* nodes at locations with a high power

```
procedure CNRRound(r)
    if controller then
        Compute expected energy at all sensors
        L_r ← compute remaining lifetime
        (L, Π) ← optimalAssignmentValue(S, P, E, F, r)
        if L_r > r then
            Broadcast new positions
            Run for duration r
            Relocate to new position according to Π
        else
            Run for duration L_r
        end if
    else
        Receive new position
        Run for duration r
        Relocate to new position according to Π
    end if
end procedure
```

Figure 6.2: Algorithm executed at the beginning of each round in a centralized setup

consumption rate, relocate to lower consumption positions. Each critical node tries to find a low power consumption rate node within its neighborhood to swap positions with. As with CNR(r), all rounds will have duration r. In each round, each node s_i with $l_i > l_{cr}$ is a critical node. Critical nodes collect the position, load, and current energy level from descendants that are at most h hops away and have not committed to switch with other critical nodes. For each candidate node c, critical node s computes $L1(c)$, the minimum of s's and c's expected lifetime without swapping, and $L2(c)$, the minimum of s's and c's expected lifetime if they swap positions. Critical node s selects candidate c^* with maximum value $L2(c)$. The swap is performed if and only if $L2(c)/L1(c) \geq f$. This is to eliminate switches that increase the network lifetime by negligible amounts. At this point c^* commits to switch with s. Figure 6.3 shows the algorithm executed by each node in each round.

```
procedure DNRRound(r, h, l_cr, f)
    if isCriticalNode then
        Compute current lifetime
        Compute minimum target lifetime
        Send REQUEST_INFO to children
    end if
    status = uncommitted

    ▷ Collect information from descendants for a period of time
    repeat
        Receive data from sender
        if data.type = STATUS_CHECK and data.target = self then
            sender status to sender
        else if data.type = SELECT and data.target = self then
            status = committed
            Perform switch with sender
            break
        else if data.type = INFO and isCriticalNode then
            ▷ If a critical node, record best candidate based on info received
            Compute expected lifetime if switched with data.candidate
            Update current bestCandidate
            if data.hops > 0 then
                data.hops = data.hops - 1
                Forward data to parent
            end if
        else if data.type = REQUEST_INFO then
            Send INFO to parent
            if data.hops > 0 then
                data.hops = data.hops - 1
                Forward data to children
            end if
        end if
    until timeout

    ▷ Select best candidate, and switch
    if isCriticalNode then
        Send(bestCandidate, STATUS_CHECK)
        Receive(bestCandidate, candidateStatus)
        if ( thencandidateStatus = uncommitted)
            Send(bestCandidate, SELECT)
            Perform switch with bestCandidate
        end if
    end if
    Run for duration r
end procedure
```

Figure 6.3: Local algorithm executed at the beginning of each round in a distributed setup

6.4 Upper Bounds on Lifetime Improvement Ratio

We now prove some upper bounds on lifetime improvement ratios for any node rotation algorithms. We first consider the 1-MaxLife problem. We then consider the general MaxLife problem. We use the following notation in our analysis. For any node rotation algorithm A and input instance I, let $L(A, I)$ denote the lifetime achieved using algorithm A on I, $L(I)$ the lifetime without node rotation, and $R_A(I) = \min_I L(A, I)/L(I)$ the lifetime improvement ratio (LIR) of A on I. Finally, let $EV(I) = \max_{i=1}^{n} e_i / \min_{j=1}^{n} e_j$ be the initial energy variance of I.

Theorem 2. *For any one round node rotation algorithm A and any input instance I $R_A(I) \leq 1 + EV(I)$.*

Proof. Let s_i denote the bottleneck node in I that determines $L(I)$; that is, $L(I) = e_i/l_i$. Clearly, $r_1 \leq L(I)$. We now observe that $r_2 \leq EV(I) \cdot L(I)$ because the node that moves to position p_i can have at most energy $EV(I) \cdot e_i$. Thus, $r_1 + r_2 \leq (1 + EV(I))L(I)$ and $R_A(I) \leq 1 + EV(I)$. □

This leads to two corollaries, one for the special case where all nodes have the same initial energy and one for multiple round rotation algorithms.

Corollary 1. *For any one round node rotation algorithm A and any input instance I with $EV(I) = 1$, $R_A(I) \leq 2$.*

Corollary 2. *For any j round node rotation algorithm A and any input instance I, $R_A(I) \leq 1 + (j-1)EV(I)$.*

We note that although our one round solution NR1 is optimal for 1-MaxLife only when $L(r_1)$ is unimodular, our simulations (Section 5.6) show that NR1's LIR is usually very close

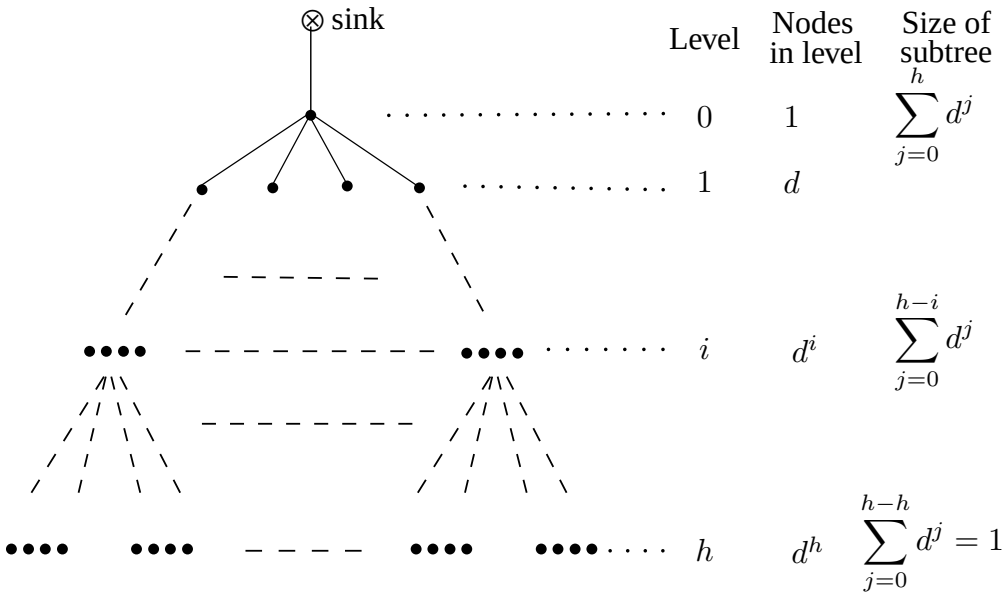

Figure 6.4: Balanced tree topology of degree $d+1$ and lowest level h

to the upper bound of 2 for input instances I with $EV(I) = 1$. Moreover, for input instances I where $EV(I) > 1$, NR1's LIR is often better than 2.

We now prove upper bounds on the $R_A(I)$ for inputs I corresponding to balanced trees of degree $d+1$ shown in Fig. 6.4.

Theorem 3. *For any node rotation algorithm A and any input I where T represents a balanced tree of degree $d+1$ and $t_{ij} = t$ for all non-zero t_{ij}, h is the lowest level of the tree where the root is at level 0, and for $1 \leq i \leq n$, $e_i = e$ and $\lambda_i = 1$, then*

$$R_A(I) \leq \frac{(d^{h+1}-1)^2}{(h(d-1)+d-2)d^{h+1}+1}$$

Proof. To prove this upper bound, we ignore the energy consumed by movement. The best solution is to then have all nodes equally share the transmission of all data. We first compute $L(I)$ which is constrained by the bottleneck node at level 0 that transmits data from the entire tree to the sink. The total number of nodes $n = \sum_{j=0}^{h} d^j$ as there are d^j nodes at

level j. It follows that

$$L(I) = \frac{e}{nt} = \frac{d-1}{d^{h+1}-1}\frac{e}{t}$$

We next compute the lifetime L^* of the WSN if we are able to perfectly share the transmission of all data among all n sensor nodes. A node at level i is the root of a subtree of size $D_i = \sum_{j=0}^{h-i} d^j$. Thus, the total amount of data transmitted each time interval is

$$D = \sum_{i=0}^{h} d^i D_i = \sum_{i=0}^{h} d^i \sum_{j=0}^{h-i} d^j = \frac{d^{h+1}(h(d-1)+d-2)+1}{(d-1)^2}$$

This implies

$$L^* = \frac{ne}{Dt} = \frac{(d^{h+1})(d-1)}{(h(d-1)+d-2)d^{h+1}+1}\frac{e}{t}$$

Dividing L^* by $L(I)$ gives us the result. □

Table 6.1 displays some of the upper bounds for different values of d and h. The improvement ratio starts at a factor of 1.8 for a tree with 3 nodes and rapidly increases with both the degree and the level.

d / h	1	2	3	4	5
2	1.80	2.88	4.59	7.45	12.36
3	2.29	4.97	11.27	26.77	66.08
4	2.78	7.74	23.08	72.99	240.82
5	3.27	11.17	41.53	164.37	679.26

Table 6.1: Upper Bounds on Lifetime Improvement Ratios in Balanced Trees

6.5 Simulation Results

In this section, we evaluate the performance of our NR1 and DNR algorithms through simulations. For comparison purposes, we also evaluate the performance of the baseline

CNR algorithm. We generated 100 networks each consisting of 100 nodes placed uniformly at random in a 150m by 150m area with the sink node chosen uniformly at random. We set the maximum communication distance to 35m, which was shown in [45] to lead to a high packet reception ratio for TelosB motes in outdoor environments. For each network, we constructed the routing tree from the sources to the sink using greedy geographic routing in which each node forwards its data to the neighbor that is closest to the sink. We note that our algorithms are applicable to network topologies generated by any routing algorithms. We set the t_{ij} and the k_{ij} values based on the energy models in [45, 13] since they are based on realistic platforms; any other energy model for communication or mobility could be used without any algorithmic change. We usually set each node's e_i to the same value typically ranging from half full to full, though in some simulations different nodes have different e_i values. For our distributed algorithm DNR, we set the local improvement factor f to 1.25 and the critical consumption rate threshold l_{cr} to 10% of the range of ratios in the network. We describe our choices for r and h below. We assess the performance of algorithms using several criteria. The main criteria is *lifetime improvement ratio*. We also assess the number of rounds required, the number of nodes that move per round, and the number of movements each node makes over the network lifetime.

6.5.1 Single Rotation Round

We first evaluate the performance of NR1 for the 1-MaxLife problem in trees. For all 100 inputs, $1.91 \leq R_{NR1}(I) \leq 1.99$, and the average value of $R_{NR1}(I) = 1.95$. We observe that the results are very close to the theoretical upper bound of 2 from Corollary 1. When we varied the starting energy level of the nodes between half full and completely full, the

average lifetime improvement ratio of NR1 increased to 2.3. We also observe that most of the nodes change their positions; on average 86 nodes relocate to new positions and 14 nodes remain at their original location. This is not too expensive since only a single rotation is performed which means the network's activity is interrupted only once.

6.5.2 Multiple Rotation Rounds

Round Duration

We now evaluate the performance of our DNR algorithm with $h = 2$ for the general MaxLife problem using our CNR(r) algorithm as a baseline. We first study the effect of varying r on the performance of both algorithms. Figure 6.5 shows the average lifetime improvement ratios for both algorithms as we increase r denoted as a fraction of the static lifetime $L(I)$. We see that DNR outperforms CNR(r) for all values of r but especially smaller r. For $r \leq 3L(I)/5$, $R_{DNR}(I) \geq 2.5 + R_{CNR}(I)$. At $r = L(I)/2$, the difference in lifetime improvement ratio is 2.5. One notable feature of DNR's performance is that the lifetime improvement ratio is relatively flat for $L(I)/5 \leq r \leq 3L(I)/5$; it takes on a maximum value of 7.4 at $r = L(I)/2$ and decreases to 7.1 at $r = 3L(I)/5$. When r is too large, the LIR of both algorithms drops because nodes stay at high consumption positions for too long.

For most of the remaining simulations, we set $r = L(I)/2$. For DNR, this almost maximizes lifetime improvement ratio while maximizing r which minimizes the number of disruptions to the network. For our comparison algorithm CNR(r), this choice maximizes the lifetime improvement ratio.

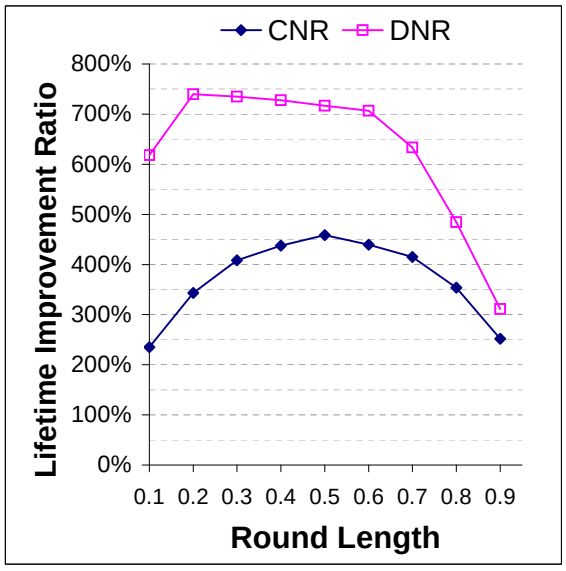

Figure 6.5: Average lifetime improvement ratios of CNR(r) and DNR as a function of r plotted as a fraction of $L(I)$

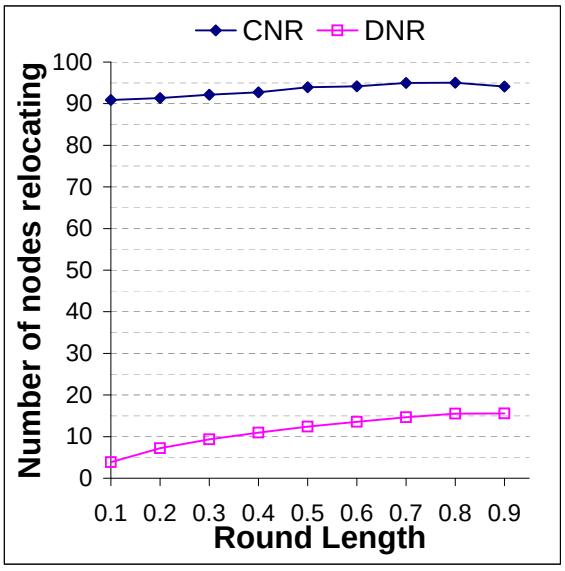

Figure 6.6: Average number of relocations of CNR(r) and DNR as a function of r plotted as a fraction of $L(I)$.

Node Relocations

We next compare the two algorithms with respect to how much the sensor nodes move. We first consider the number of relocations per round. As we can see from Fig. 6.6, DNR outperforms CNR again. In particular, with CNR, more than 90% of the nodes relocate in each round whereas with DNR, the average number of relocating nodes stays below 16%. These nodes are usually different in each round as 96% of the nodes move between 0 and 3 times overall and the remaining 4% move up to 6 times during their lifetime. For DNR, most of the energy available at each node is consumed by communication rather than movement: 85% of the nodes spend at least 80% of their energy on communication and no node spends more than 35% of its energy on movement. We see observe that node relocations increase only slightly for DNR as round duration r increases because more nodes become critical in each round due to more time at high consumption locations.

Distributed Approach and Hop Distance Parameter h

We now analyze the effect of the hop distance parameter h on the performance of DNR. We plot the complementary cumulative distribution function (CCDF) of the lifetime improvement ratio for DNR with h set to 1, 2, and 4 in Fig. 6.7. The CCDF gives us the probability that the lifetime improvement ratio exceeds a given threshold. For comparison, we also plot the CCDF of CNR. From this data, we see that setting $h = 2$ is sufficient to achieve excellent performance as the CCDF for $h = 2$ is almost identical to that of $h = 4$. In both cases, 93% of the topologies have lifetime improvement ratios of at least 400% and more than 50% of topologies have lifetime improvement ratios over 700%. With $h = 1$, DNR is much less effective; the number of nodes taking turns transmitting from the high consumption position may be too low. In the remaining simulations, we set $h = 2$ for DNR.

Lifetime Improvement Ratio Increase per Round

We now assess how much effect each round has on the lifetime improvement ratio. Specifically, if we stop node rotations after round n, what will the lifetime be? For both CNR and DNR, we obtain a lifetime improvement ratio that is essentially linear in the number of rounds with each round increasing the lifetime improvement ratio by between 40 and 50%. This analysis shows that both algorithms are effective in increasing the lifetime improvement ratio but that DNR is more effective in minimizing distance moved and maintaining a reserve of energy rich nodes for later rounds. This is why DNR outperforms CNR which moves 93% of the nodes in each round.

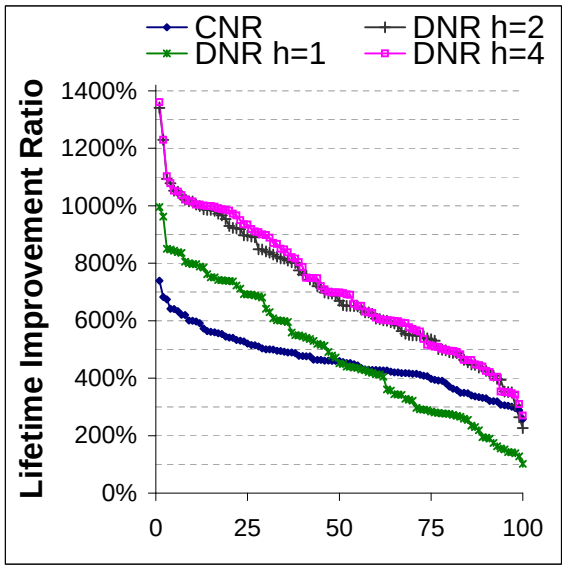

Figure 6.7: CCDF of lifetime improvement ratio of CNR and DNR with $r = L(I)/2$ and $h = 1$, 2, and 4.

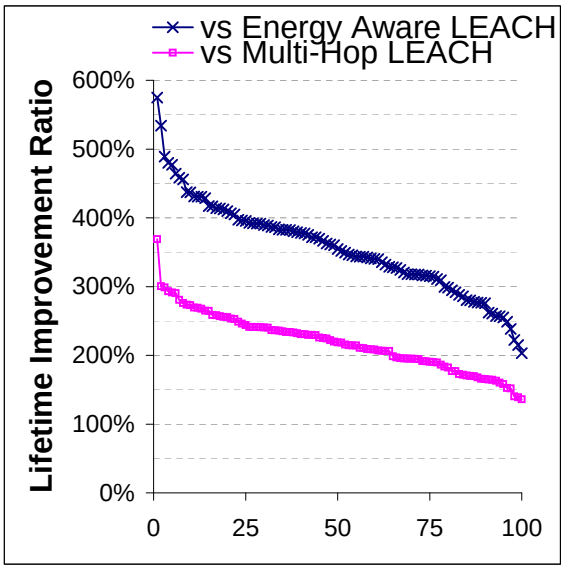

Figure 6.8: CCDF of the lifetime comparison ratios of DNR versus LEACH and multihop LEACH.

Comparison with Existing Approaches

In this section, we compare DNR to existing approaches. We consider only approaches that do not change the positions of where nodes are placed. This rules out existing mobile relay approaches and leaves us with non-mobility approaches like LEACH [19] that rotate the roles of different nodes by periodically changing the topology of the network but not modifying any node positions. Specifically, we compare DNR with LEACH [19] as it serves as the base for several other clustering algorithms that seek to increase network lifetime and (2) multihop LEACH [33], an improved variation that uses multihop transmissions between cluster heads. Both LEACH approaches assume that data is compressed before being transmitted while DNR does not. To compare all approaches in a similar setting, we run them all without data compression.

We compare DNR to LEACH and multihop LEACH on an input instance I by comput-

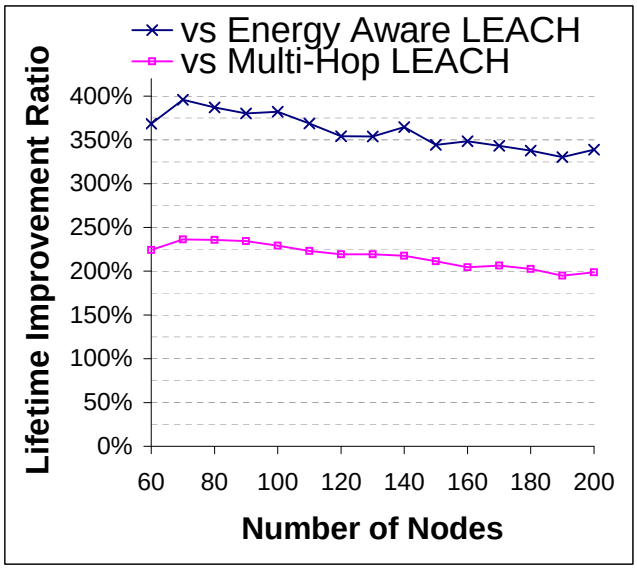

Figure 6.9: Average improvement ratio of our distributed approach DNR with respect to energy aware LEACH and multi-hop LEACH as the number of nodes in the network increases

ing the *lifetime comparison ratio* $R_{DNR}(I)/R_A(I)$ where A is either LEACH or multihop LEACH. Figure 6.8 shows the complementary cumulative distribution function of both lifetime comparison ratios. First, we note that DNR outperforms both LEACH variations for every topology, attaining lifetimes between 2 and 5.75 times better than LEACH and between 1.4 and 3.7 times better than multihop LEACH. Additionally, we observe that DNR needs many fewer rounds than both LEACH variations. On average, DNR needs 8 rounds of rotations whereas the average number of rounds is 1800 for LEACH and 2000 for multihop LEACH. We also note that the round duration r used for the LEACH approaches was 20% of the r used by DNR as using the same r resulted in much lower lifetime improvements ratios for LEACH.

We now compare the performance of all three approaches as the density of the network varies. We varied the number of nodes between 60 and 200 in increments of 10. We generated a new set of topologies in which we varied the number of nodes between 60 and 200 in incre-

ments of 10. For each size, we randomly generated 75 networks. We observe in Fig. 6.9 that the average improvement ratio of our distributed approach with respect to both LEACH variations decreases slowly as the density of the network increases. Compared to energy aware LEACH, our distributed approach yields lifetimes between 3.3 and 4.0 times better. Compared to multihop LEACH, our approach maintains an average improvement ratio between 2 and 2.3 times better. We also observe that as the density of the network increases, the number of rounds performed increases significantly for both LEACH approaches (from 1600 to 2200 rounds in energy aware LEACH and from 1600 to 3000 rounds in multihop LEACH) whereas DNR requires on average only two more rounds.

6.6 Summary

In this chapter, we consider the problem of maximizing the lifetime of mobile WSNs. We exploit the mobility of nodes to mitigate differential power consumption by having nodes take turns in high power consumption positions without modifying the existing communication topology. We present efficient algorithms for both the single round and the general multiple round MaxLife problem. This approach is very different than other schemes such as data mules in that all nodes expend relatively little energy on movement and move only a few times during the network lifetime. Our simulations show that our algorithms can improve average lifetime by more than a factor of 7 and that our algorithms outperform existing non-mobility approaches for mitigating differential power consumption to prolong network lifetime.

CHAPTER 7

Conclusion

In this thesis, we leverage node mobility to improve the energy efficiency of wireless sensor networks. We propose several approaches for optimizing different performance metrics. In this chapter, we first summarize our proposed approaches. Then, we discuss some possible open problems.

7.1 Summary

We propose to use low-cost disposable mobile nodes to reduce the energy consumption in wireless sensor networks and increase the networks lifetime. Our proposed approaches are holistic as they take into consideration the energy consumed by both mobility of relays and wireless transmissions and receptions. Most previous work ignored the energy consumed by moving mobile relays. Unlike most previous work, for each problem we consider, we carefully compute the optimal target location of each node. Relocating to the optimal position showed its superiority over existing approaches that use an approximate location.

We first consider a new problem, MMRC, maximizing the data gathering capacity of

hybrid wireless sensor networks consisting of mainly static nodes and a few mobile nodes. We study how to exploit a small number of mobile nodes to improve the data gathering capacity of the network. We examine two cases in particular, when one and then two mobile nodes are allowed to join a single link. We present optimal and approximate solutions to four variants of MMRC. In most variants (tree topologies), our algorithms improved the system lifetime by a factor of 2 or more. For the star topology, it increased system lifetime by a factor of 1.5. Our distributed protocols are almost as effective at improving the data gathering capacity, particularly as the number of mobile relay nodes increases. Our simulations also show that these protocols quickly converge on a final solution with little messaging overhead. Moreover, our simulations show that it is enough to consider a single mobile relay per link to get the highest improvements.

Second, we study the problem of minimizing the total energy consumption in a wireless network. For this problem, we consider networks entirely consisting of mobile nodes and utilize the mobility of all nodes to reduce the energy consumption. We design a framework for minimizing the total energy consumption in a sensor network consisting in both constructing low consumption transmission routes and repositioning the nodes to their optimal positions. We first identify a tree construction strategy that results in an optimal configuration in a static environment. Then we develop an algorithm that uses this optimal configuration and iteratively refines it by inserting new relay nodes into the tree. Moreover, we develop a second iterative algorithm that improves the given configuration by relocating nodes to new positions. We show that this algorithm converges to the optimal configuration for the given transmission routes. Our simulations show that our algorithm can reduce the total energy consumption by an average of 45%.

Finally, we consider the problem of maximizing the network lifetime. Again we exploit the mobility of all nodes to balance the energy consumption across the network. We propose to relocate nodes rich in energy to locations with higher consumption rates and conversely relocate nodes low in energy to locations with low consumption rates. We address this problem for single and multiple rounds of rotations. We show that a single rotation improves the lifetime by a factor of almost 2. Our simulations show that our algorithms for multiple rounds of rotations can improve the average lifetime by more than a factor of 7 and that they outperform existing non-mobility approaches for mitigating differential power consumption to prolong network lifetime in both overhead and lifetimes achieved.

7.2 Open problems

In our solutions for maximizing the data gathering capacity, we assumed that the routing tree connecting the static nodes to the sink is predetermined and we proposed an insertion approach to improve the given tree. Likewise, in our solutions for the lifetime maximization problem, we maintained the same tree throughout the process of node rotation. These solutions significantly improve the performance of the given tree. However, in some cases, the initial routing structure may not be defined or the application may not dictate any route constraints. These cases may benefit from joint optimization of the routing structure and the nodes' locations to fully exploit mobility of nodes.

Throughout this study, we adopted one particular definition of network lifetime which is the time until the first node dies. This definition coincides with the lifetime of a configuration as the death of a single node entails the need for a new configuration. However, some applications may continue to function properly even after a small fraction of the nodes fails.

It is interesting to extend our solutions to such applications where the lifetime is defined by several criteria such as the fraction of the network alive and some minimum coverage requirements.

APPENDICES

APPENDIX A

Proofs of Claims and Theorems

Claim 1. *$g_r(\alpha)$ is unimodal*

Proof. Recall that the function $g_r(\alpha)$ is defined as the value of the maximum message that can be transmitted by r when it moves to $pos_r(\alpha)$. We define and $m_r^\alpha(t)$ as the maximum message r can transmit to s_2 when it moves to point t along $r\alpha$. We also define γ to be the angle between $r\alpha$ and rs_2 and δ to be the distance traveled by r to get to point t as shown in Figure A.1. Since both γ and δ uniquely determine the point t, we redefine $m_r^\alpha(t)$ as a function of δ as follows:

$$\begin{aligned} M_\gamma(\delta) &= \frac{e_r - k\delta}{a + b((ts_2)^2} \\ &= \frac{e_r - k\delta}{a + b(\delta^2 + d_2^2 - 2\delta d_2 \cos(\gamma))} \end{aligned}$$

We define δ_l to be the maximum distance r can travel before it consumes all its energy by

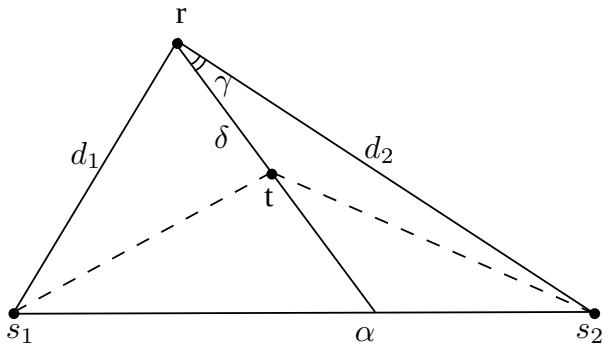

Figure A.1: $g_r(\alpha)$ is unimodal

moving. We have $\delta_l = \frac{e_t}{k}$. We compute the derivative of M_γ with respect to δ. We get:

$$\begin{aligned} M'_\gamma(\delta) &= -\frac{k}{a + b(\delta^2 + d_2^2 - 2\delta d_2 \cos(\gamma))} - \frac{(e_r - k\delta)b(2\delta - 2d_2\cos(\gamma))}{(a + b * (\delta^2 + d_2^2 - 2\delta d_2 \cos(\gamma)))^2} \\ &= \frac{kb\delta^2 - 2e_r b\delta - k(a + bd_2^2) + 2e_r b d_2 \cos(\gamma)}{(a + b(\delta^2 + d_2^2 - 2d_2\delta\cos(\gamma)))^2} \end{aligned}$$

So $M'_\gamma(\delta)$ has the same sign as $E = kb\delta^2 - 2e_r b\delta - k(a + bd_2^2) + 2e_r b d_2 \cos(\gamma)$.

This is a second degree polynomial, we study its variation as follows. Let Δ be its determinant.

$$\begin{aligned} \Delta &= (-2e_r b)^2 - 4(kb)(-k(a + bd_2^2) + 2e_r b d_2 \cos(\gamma)) \\ &= 4e_r^2 b^2 + 4k^2 b(a + bd_2^2) - 8kb^2 e_r d_2 \cos(\gamma) \end{aligned}$$

$$\begin{aligned} &\Delta > 0 \\ &\Leftrightarrow \cos(\gamma) < \frac{e_r^2 b + k^2(a + bd_2^2)}{2kbd_2 e_r} \\ &\Leftrightarrow \cos(\gamma) < \frac{e_r^2 + k^2 d_2^2}{2kd_2 e_r} + \frac{ka}{2bd_2 e_r} \end{aligned}$$

But $\frac{e_r^2 + k^2 d_2^2}{2kd_2 e_r} > 1$ and $\frac{ka}{2bd_2 e_r} > 0 \quad \forall \quad e_r, k, d_2 > 0$

So $\Delta > 0 \Leftrightarrow \cos(\gamma) < 1 + v^+$, for some $v^+ > 0$ and consequently

$$\Delta > 0 \quad \forall \quad e_r, k, d_2, a, b > 0, 0 < \gamma < \pi \tag{A.1}$$

The roots of E are:

$$\begin{aligned}\delta^- &= \frac{e_r}{k} - \frac{\sqrt{\Delta}}{2kb} \\ \delta^+ &= \frac{e_r}{k} + \frac{\sqrt{\Delta}}{2kb} > \delta_l\end{aligned} \tag{A.2}$$

In this case, the function M_γ varies as follows with δ:

δ	$-\infty$		δ^-		δ^+		$+\infty$
$(\delta - \delta^-)$	$-$		$+$		$+$		
$(\delta - \delta^+)$	$-$		$-$		$+$		
$kb(\delta - \delta^-)(\delta - \delta^+)$	$+$		$-$		$+$		
M_γ	$\nearrow$		$\searrow$		$\nearrow$		

So the maximum of M_γ is at δ^- if $\delta >= 0$ and at 0 otherwise. We now study how the maximum at each δ^- varies has γ varies between 0 and π. We define this function g_r^γ as:

$$\begin{aligned}g_r^\gamma(\gamma) &= \frac{e_r - k\delta^-(\gamma)}{a + b(\delta^-(\gamma)^2 + d_2^2 - 2d_2\delta^-(\gamma)\cos(\gamma))} \\ &= \frac{k^2\sqrt{e_r^2b^2 + k^2ab + k^2b^2d_2^2 - 2v_1}}{v_2}\end{aligned}$$

where

$$\begin{aligned}v_1 &= kbe_rd_2\cos(\gamma) \\ v_2 &= e_r^2b^2 - e_rb\sqrt{e_r^2b^2 + k^2ab + k^2b^2d_2^2 - 2v_1} + k^2ab + k^2b^2d_2^2 - 2v_1 \\ &\quad + d_2\cos(\gamma)k\sqrt{e_r^2b^2 + k^2ab + k^2b^2d_2^2 - 2v_1}\end{aligned}$$

We compute the derivative $g_r^{\gamma'}(\gamma)$ which corresponds to:

$$g_r^{\gamma'}(\gamma) = \frac{2d_2}{k}\frac{T}{B^2}$$

with

$$T = -\sin(\gamma)(e_r^2b^2 + k^2ab + k^2b^2d_2^2 - 2v)(e_rb - \sqrt{e_r^2b^2 + k^2ab + k^2b^2d_2^2 - 2v})$$

So $g_r^{\gamma'}(\gamma)$ has the same sign as T. We analyze the sign of $g_r^{\gamma'}(\gamma)$ for $0 \leq \gamma \leq \pi$ as follows. We observe that T is the product of three factors:

1. $\sin(\gamma) > 0, \quad \forall \quad 0 \leq \gamma \leq \pi$
2. $e_r^2b^2 + k^2ab + k^2b^2d_2^2 - 2v > 0 \Leftrightarrow \cos(\gamma) \leq \dfrac{e_rb + k^2a + k^2bd_2^2}{2kbe_rd_2}$

   but we can show that $\dfrac{e_rb + k^2a + k^2bd_2^2}{2kbe_rd_2} < 1 \quad \forall \quad e_r \geq 0, b \geq 0, k \geq 0, a \geq 0$

   so $e_r^2b^2 + k^2ab + k^2b^2d_2^2 - 2v > 0 \quad \forall \quad 0 \leq \gamma \leq \pi$
3. $e_rb - \sqrt{e_r^2b^2 + k^2ab + k^2b^2d_2^2 - 2v}$: this becomes 0 when $\cos(\gamma) = \dfrac{k(a + bd_2^2)}{d_2be_r}$. We distinguish two cases:

   - Case 1:

   $$\frac{k(a + bd_2^2)}{d_2be_r} < 1 \Leftrightarrow e2 > \frac{k(a + bd_2^2)}{db}$$

   $$\text{for } \gamma_0 = \arccos(\frac{k(a + bd_2^2)}{d_2be_r}), e_r^2b^2 + k^2ab + k^2b^2d_2^2 - 2v = 0$$

   $$e_r^2b^2 + k^2ab + k^2b^2d_2^2 - 2v > 0 \quad \forall \quad 0 < \gamma < \gamma_0$$

   $$e_r^2b^2 + k^2ab + k^2b^2d_2^2 - 2v < 0 \quad \forall \quad \gamma_0 < \gamma < \pi$$

   - Case 2:

   $$\frac{k(a + bd_2^2)}{d_2be_r} > 1 \Leftrightarrow e_r < \frac{k(a + bd_2^2)}{sb}$$

   $$\Rightarrow \cos(\gamma) < \frac{k(a + bd_2^2)}{d_2be_r} \quad \forall \quad 0 \leq \gamma \leq \pi$$

   $$e_r^2b^2 + k^2ab + k^2b^2d_2^2 - 2v < 0 \quad \forall \quad 0 < \gamma < \pi$$

The following tables represent the variation of function $g_r^{\gamma}(\gamma)$ for $0 < \gamma < \pi$:

Case 1

δ	0	γ_0	∞
-1	$-$		$-$
$\sin(\gamma)$	$+$		$+$
$e_r^2b^2 + k^2ab + k^2b^2d_2^2 - 2v$	$+$		$-$
$e_rb - \sqrt{e_r^2b^2 + k^2ab + k^2b^2d_2^2 - 2v}$	$+$		$+$
$g_r^{\gamma'}(\gamma)$	$-$		$+$
$g_r^{\gamma}(\gamma)$	$\searrow$		$\nearrow$

Case 2

δ	0	∞
-1	$-$	
$\sin(\gamma)$	$+$	
$e_r^2b^2 + k^2ab + k^2b^2d_2^2 - 2v$	$-$	
$e_rb - \sqrt{e_r^2b^2 + k^2ab + k^2b^2d_2^2 - 2v}$	$+$	
$g_r^{\gamma'}(\gamma)$	$-$	
$g_r^{\gamma}(\gamma)$	$\searrow$	

As γ increases, α decreases. So when $g_r^{\gamma}(\gamma)$ is increasing, $g_r(\alpha)$ is decreasing and conversely, when when $g_r^{\gamma}(\gamma)$ is decreasing, $g_r(\alpha)$ is increasing. Therefore the function $g_r(\alpha)$ is unimodal.

□

Claim 2. *$h_b(\alpha)$ is decreasing between s_1 and s_2.*

Proof. Given a direction $r\alpha$, $h_b(\alpha)$ is the largest message that s_1 can transmit to r when it moves along $r\alpha$. This corresponds to r moving to the closest point on $r\alpha$ to s_1, which is the intersection of the perpendicular line to $r\alpha$ taken from s_1. We call this point h_α. We now show that the distance between s_1 and $r\alpha$ increases as α increases. We consider two cases: when $x_1 \leq x_r \leq x_2$ and when $x_r \leq x_1 \leq x_2$.

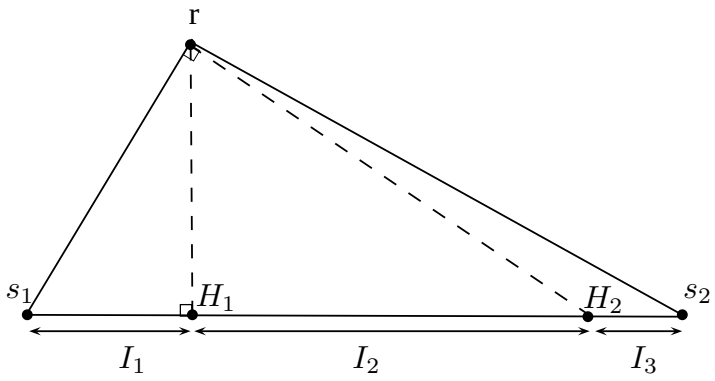

Figure A.2: Subdivision of s_1s_2 into 3 intervals.

Case I: $x_1 \leq x_r \leq x_2$

Let H_1 be the intersection of s_1s_2 and the perpendicular to it from r, and H_2 the intersection between s_1s_2 and the perpendicular to s_1r from r. We divide the segment s_1s_2 into three intervals: $I_1 = [s_1, H_1],\ I_2 = [H_1, H_2],\ I_3 = [H_2, s_2]$ as shown by Figure A.2.

For any point $\alpha \in I_1$, h_α falls outside the triangle s_1rs_2. Since we only consider points inside the triangle, the point maximizing $h_b(\alpha)$ is α. The distance $s_1\alpha$ increases as α moves towards H_1. So $h_b(\alpha)$ is decreasing in I_1.

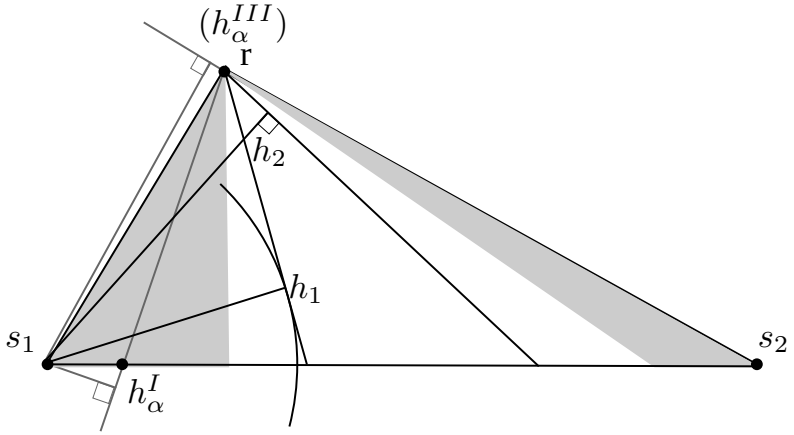

Figure A.3: In I_1, h_α is on the line s_1s_α. In I_2, s_1h_α increases with s_α. In I_3, h_α coincides with r.

Consider two points $s_{\alpha_1} \in I_2$ and $s_{\alpha_2} \in I_2$ such that $x_{\alpha_1} < x_{\alpha_2}$ as shown in Figure A.3.

Let $s_1h_{\alpha_1}$ be the perpendicular segment from s_1 to rs_{α_1}. Let C be the circle of center s_1 and radius $s_1h_{\alpha_1}$. The line rh_{α_1} is tangent to C from r. Since $\alpha_1 < \alpha_2$, the line rh_{α_2} does not intersect C so $s_1h_{\alpha_2} > radius(C)$ and $s_1h_{\alpha_2} > s_1h_{\alpha_1}$. So $h_b(\alpha)$ is decreasing in interval I_2.

For any point $\alpha \in I_3$, h_α falls outside the triangle s_1rs_2. In this case, the point maximizing $h_b(\alpha)$ is s_2. The distance s_1r remains constant as α moves from H_2 towards s_3.

Case II: $x_r \leq x_1 \leq x_2$

The same argument as case I, interval I_2 applies in this case and the optimal transmission distance for s_1 increases with α. So $h_b(\alpha)$ is decreasing between s_1s_2. □

Claim 3. *$f(\alpha)$ is unimodal*

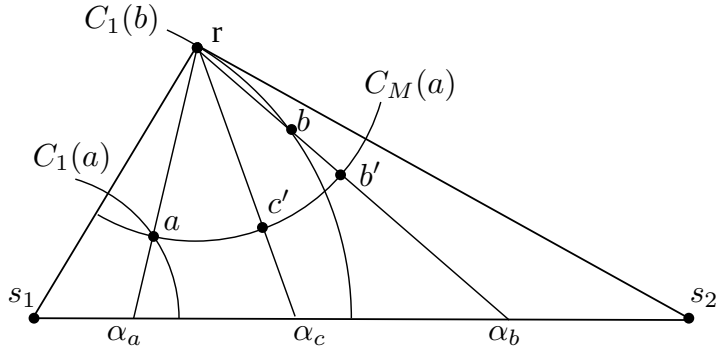

Figure A.4: Any point c between a and b such that $m(a) > m(b)$ satisfies the condition $m(c) > m(b)$

Proof. Consider directions along three points $\alpha_a < \alpha_c < \alpha_b$ such that $\overline{M}(\alpha_a) = f(\alpha_a) > f(\alpha_b) = \overline{M}(\alpha_b)$. We consider the case when $\alpha_a \in I_1$ but the remaining cases (for $\alpha_a \in I_2$ and $\alpha_a \in I_3$) follow similar geometric arguments.

Let a be the point of segment $r\alpha_a$ where $f(\alpha_a)$ reaches its maximum and b the point along

segment $r\alpha_b$ where $f(\alpha_b)$ reaches its maximum. We define $C_1(a)$ to be the circle of center s_1 and radius s_1a, $C_1(b)$ the circle of center s_1 and radius s_1b, and $C_M(a)$ the circle of center r and radius ra.

We now show that if $\overline{M}(\alpha_c) = f(\alpha_c)$ for some point c on $r\alpha_c$ then $f(\alpha_c) > f(\alpha_b)$ and consequently $\overline{M}(\alpha_a) > \overline{M}(\alpha_c) > \overline{M}(\alpha_b)$

First, we show the position of point b relative to point a. We define the point b' to be the intersection of circle $C_M(a)$ with segment $r\alpha_b$. At point b', r traveled the same distance as at point a but $b's_2 < as_2$, so

$$m_2(b') > m_2(a) \tag{A.3}$$

But since b is the optimal point on the segment $r\alpha_b$, $f(\alpha_b) > \min\{m_1(b'), m_2(b')\}$. On the other hand, $f(\alpha_a) > f(\alpha_b)$, therefore

$$f(\alpha_a) > \min\{m_1(b'), m_2(b')\}$$

This along with (A.3) $\Rightarrow$

$$f(\alpha_a) > m_1(b')$$

So $m_1(b') = \min\{m_1(b'), m_2(b')\}$. But since $f(\alpha_b) > m_1(b')$ then $m_1(b) > m_1(b')$. So the point b must be within the circle of center s_1 and radius s_1b' as shown in Figure A.4.

We now show that the function f never increases after it starts decreasing. Let c' be the intersection of segment $r\alpha_c$ with circle $C_M(a)$. Point c' is on the circle $C_M(a)$ and is closer to s_2 than a is, so:

$$m_2(c') > m_2(a) > m_2(b)$$

Besides, c' is on the inside of circle $C_1(b)$ so:

$$m_1(c') > m_1(b)$$

$$\Rightarrow \min\{m_1(c'), m_2(c')\} > m_1(b) = m_2(b)$$

But

$$f(\alpha_c) > \min\{m_1(c'), m_2(c')\} \quad \text{since } c \text{ is the optimal point on } r\alpha_c$$

$$\Rightarrow f(\alpha_c) > f(\alpha_b)$$

□

Claim 4. *$g_b(\alpha)$ is continuous*

Proof. We prove that the function $g_b(\alpha)$ is continuous by showing it is differentiable for any α in s_1s_2. First, we define $\hat{\alpha}$ as the angle $\widehat{s_1rc}$ and θ as the angle $\widehat{s_1rs_2}$. We note that θ is a constant. We then redefine $g_b(\alpha)$ in terms of $\hat{\alpha}$ as:

$$g_b(\hat{\alpha}) = e_1/(a + bu(\hat{\alpha})^2)$$

with $u(\hat{\alpha})$ being the transmission distance for s_1 to r at its new position (*i.e.* the distance s_1t in Figure A.1). We note that $g_b(\alpha)$ and $g_b(\hat{\alpha})$ are equal and $g_b(\alpha)$ is continuous and differentiable whenever $g_b(\hat{\alpha})$ is. Consider triangle $\widehat{s_1rt}$, we have

$$u(\hat{\alpha})^2 = \delta^2(\gamma) + d_1^2 - 2\delta(\gamma)d_1cos(\hat{\alpha})$$

So

$$g_b(\hat{\alpha}) = e_1/(a + b(\delta^2(\gamma) + d_1^2 - 2\delta(\gamma)d_1\cos(\hat{\alpha})))$$

$$= e_1/(a + b(\delta^2(\theta - \hat{\alpha}) + d_1^2 - 2\delta(\theta - \hat{\alpha})d_1\cos(\hat{\alpha})))$$

Using the value of $\delta(\gamma)$ in (A.2), and simplifying the result, we get:

$$g_b(\hat{\alpha}) = e_1k^2/\Big(2k^2a + 2e_r^2b - 2e_r\sqrt{b(e_r^2b + k^2a + k^2bd_2^2 - 2kbe_rd_2\cos(\hat{\alpha}))}+$$

$$k^2bd_2^2 - 2kbe_rd_2cos(\hat{\alpha}) + d_1^2k^2b - 2d_1\cos(\theta - \hat{\alpha})kbe_r+$$

$$2d_1\cos(\theta - \hat{\alpha})k\sqrt{b(e_r^2b + k^2a + k^2bd_2^2 - 2kbe_rd_2\cos(\hat{\alpha}))}\Big)$$

and its derivative with respect to $\hat{\alpha}$ is:

$$g_b'(\hat{\alpha}) = -e_1b\Big(- 2\frac{(\frac{e_r}{k} - \frac{\sqrt{\Delta}}{kb})be_rd_2\sin(\hat{\alpha})}{\sqrt{\Delta}} + 2\frac{be_rd_2\sin(\hat{\alpha})d_1\cos(\theta - \hat{\alpha})}{\sqrt{\Delta}}$$

$$+2(\frac{e_r}{k} - \frac{\sqrt{\Delta}}{kb})d_1\sin(\theta - \hat{\alpha})\Big)$$

$$/$$

$$\Big(a + b((\frac{e_r}{k} - \frac{\sqrt{\Delta}}{kb})^2 + d_1^2 - 2(\frac{e_r}{k} - \frac{\sqrt{\Delta}}{kb})d_1\cos(\theta - \hat{\alpha}))\Big)^2$$

The derivative $g_b'(\hat{\alpha})$ exists when the following two conditions hold:

1. $\sqrt{\Delta} \neq 0$ which is true according to (A.1) in the proof of claim (1)
2. $a + b((\frac{e_r}{k} - \frac{\sqrt{\Delta}}{kb})^2 + d_1^2 - 2(\frac{e_r}{k} - \frac{\sqrt{\Delta}}{kb})d_1\cos(\theta - \hat{\alpha})) \neq 0$: but $a > 0$ and $(\frac{e_r}{k} - \frac{\sqrt{\Delta}}{kb})^2 + d_1^2 - 2(\frac{e_r}{k} - \frac{\sqrt{\Delta}}{kb})d_1\cos(\theta - \hat{\alpha}) = u > 0$

So $g_b(\hat{\alpha})$ is differentiable and consequently continuous and the same applies to $g_b(\alpha)$.

□

Claim 5. *$h_r(\alpha)$ is continuous*

Proof. We prove that the function $h_r(\alpha)$ is continuous by showing it is differentiable. As previously defined, $\hat{\alpha}$ is the angle $\widehat{s_1 r \alpha}$, δ the distance traveled by r, and u the distance between s_1 and r at its new position. We additionally define w as the distance between the new position of r and s_2. We consider the three intervals I_1, I_2, I_3 of Figure A.2.

Case I: $\alpha \in I_1$

We define β to be the angle between $s_1 r$ and rH_1, and h to be the distance rH_1.

$$\begin{aligned}
\delta &= \frac{h}{\cos(\beta - \hat{\alpha})} \\
w^2 &= \delta^2 + d_2^2 - 2\delta d_2 \cos(\gamma) \\
&= \frac{h^2}{\cos(\beta - \hat{\alpha})^2} + d_2^2 - 2\frac{h}{\cos(\beta - \hat{\alpha})} d_2 \cos(\theta - \hat{\alpha}) \\
h_r(\alpha) &= h_r(\hat{\alpha}) = \frac{e_r - k\delta}{a + bw^2} \\
&= \frac{e_r - k\dfrac{h}{\cos(\beta - \hat{\alpha})}}{a + b(\dfrac{h^2}{\cos(\beta - \hat{\alpha})^2} + d_2^2 - 2\dfrac{h}{\cos(\beta - \hat{\alpha})} d_2 \cos(\theta - \hat{\alpha}))}
\end{aligned}$$

And the derivative of $h_r(\hat{\alpha})$ is:

$$\begin{aligned}
h_r'(\hat{\alpha}) = &\frac{kh \sin(\beta - \hat{\alpha})}{\cos(\beta - \hat{\alpha})^2 \Big(a + b(\dfrac{h^2}{\cos(\beta - \hat{\alpha})^2} + d_2^2 - 2\dfrac{h}{\cos(\beta - \hat{\alpha})} d_2 \cos(\theta - \hat{\alpha}))\Big)} \\
&- \frac{b(e_r - \dfrac{kh}{\cos(\beta - \hat{\alpha})})}{\Big(a + b(\dfrac{h^2}{\cos(\beta - \hat{\alpha})^2} + d_2^2 - 2\dfrac{h}{\cos(\beta - \hat{\alpha})} d_2 \cos(\theta - \hat{\alpha}))\Big)^2} * \\
&(-2\frac{h^2 \sin(\beta - \hat{\alpha})}{\cos(\beta - \hat{\alpha})^3} + 2\frac{h d_2 \cos(\theta - \hat{\alpha}) \sin(\beta - \hat{\alpha})}{\cos(\beta - \hat{\alpha})^2} + 2\frac{h d_2 \sin(\theta - \hat{\alpha})}{\cos(\beta - \hat{\alpha})})
\end{aligned}$$

The derivative exists when the following conditions hold:

1. $\cos(\beta - \hat{\alpha}) \neq 0$

2. $a + b(\frac{h^2}{\cos(\beta - \hat{\alpha})^2} + d_2^2 - 2\frac{h}{\cos(\beta - \hat{\alpha})} d_2 \cos(\theta - \hat{\alpha})) \neq 0$

But since $s_1 r$ and rH_1 are not parallel, $\beta < \frac{\pi}{2}$ and $\cos(\beta - \hat{\alpha}) > 0, \ \forall \ 0 \leq \hat{\alpha} < \beta$. So the first condition is satisfied. On the other hand,

$$a + b(\frac{h^2}{\cos(\beta - \hat{\alpha})^2} + d_2^2 - 2\frac{h}{\cos(\beta - \hat{\alpha})} d_2 \cos(\theta - \hat{\alpha})) = a + bw^2 > 0$$

So the second condition is also satisfied and consequently the derivative $h_r'(\hat{\alpha})$ exists in this case and $h_r(\hat{\alpha})$ is continuous and so is $h_r(\alpha)$.

Case II: $\alpha \in I_2$

In this case,

$$\begin{aligned}
\delta &= d_1 \cos(\hat{\alpha}) \\
u &= d_1 \sin(\hat{\alpha}) \\
w^2 &= \delta^2 + d_2^2 - 2\delta d_2 \cos(\gamma) \\
&= d_1^2 \cos(\hat{\alpha})^2 + d_2^2 - 2 d_1 d_2 \cos(\hat{\alpha}) \cos(\theta - \hat{\alpha}) \\
h_r(\hat{\alpha}) &= \frac{e_r - k\delta}{a + bw^2} \\
&= \frac{e_r - k d_1 \cos(\hat{\alpha})}{a + b(d_1^2 \cos(\hat{\alpha})^2 + d_2^2 - 2 d_1 d_2 \cos(\hat{\alpha}) \cos(\theta - \hat{\alpha}))}
\end{aligned}$$

The derivative of $h_r(\hat{\alpha})$ is:

$$h_r'(\hat{\alpha}) = \frac{k d_1 \sin(\hat{\alpha})}{a + b(d_1^2 \cos(\hat{\alpha})^2 + d_2^2 - 2 d_1 d_2 \cos(\hat{\alpha}) \cos(\theta - \hat{\alpha}))} - \frac{(e_r - k d_1 \cos(\hat{\alpha})) b(-2 d_1^2 \cos(\hat{\alpha}) \sin(\hat{\alpha}) + 2 d_1 d_2 \sin(\hat{\alpha}) \cos(\theta - \hat{\alpha}) - 2 d_1 d_2 \cos(\hat{\alpha}) \sin(\theta - \hat{\alpha}))}{\Big(a + b(d_1^2 \cos(\hat{\alpha})^2 + d_2^2 - 2 d_1 d_2 \cos(\hat{\alpha}) \cos(\theta - \hat{\alpha}))\Big)^2}$$

The derivative exists when $a + b(d_1^2 \cos(\hat{\alpha})^2 + d_2^2 - 2 d_1 d_2 \cos(\hat{\alpha}) \cos(\theta - \hat{\alpha})) \neq 0$. But this is

equal to $a + bw^2 > 0$. So h_r is differentiable and continuous.

Case III: $\alpha \in I_3$

In this case, $pos_b(\alpha) = r$ and the function $h_r(\alpha)$ is constant and continuous.

As a result of all three cases, $h_r(\alpha)$ is continuous in each interval. Also, the value of h_r at point H_1 is the same in both intervals I_1 and I_2 and the value at point H_2 is the same in I_2 and I_3. So h_r is continuous over the interval $I_1 \cup I_2 \cup I_3$, ie, for α between s_1 and s_2. □

Theorem 4. *The optimal position for node s_i when it moves by itself is given by*

$$x_i = p_i + \frac{-B_x(\sqrt{B_x^2 + B_y^2} \pm k)}{A\sqrt{B_x^2 + B_y^2}}$$
$$y_i = q_i + \frac{-B_y(\sqrt{B_x^2 + B_y^2} \pm k)}{A\sqrt{B_x^2 + B_y^2}}$$

where

$$A = 2b(\sum_{s_l \in S(s_i)} m_l + m_i)$$

$$B_x = 2b(\sum_{s_l \in S(s_i)} m_l + m_i)p_i - 2b(\sum_{s_l \in S(s_i)} m_l x_l + m_i x_d)$$

$$B_y = 2b(\sum_{s_l \in S(s_i)} m_l + m_i)q_i - 2b(\sum_{s_l \in S(s_i)} m_l y_l + m_i y_d)$$

Proof. We compute the derivatives $\frac{\delta C_i}{dx_i}$ and $\frac{\delta C_i}{dy_i}$ as follows.

$$
\begin{aligned}
C_i = & \sum_{s_l \in S(s_i)} \left(am_l + bm_l \left\| u_i - u_l \right\|^2 \right) \\
& + am_i + bm_i \left\| u_i - u_d \right\|^2 + k \left\| u_i - o_i \right\| \\
= & \sum_{s_l \in S(s_i)} \left(am_l + bm_l (x_i - x_l)^2 + bm_l (y_i - y_l)^2 \right) \\
& + am_i + bm_i (x_i - x_d)^2 + bm_i (y_i - y_d)^2 \\
& + k\sqrt{(x_i - p_i)^2 + (y_i - q_i)^2} \\
\frac{\delta C_i}{dx_i} = & 2b \sum_{s_l \in S(s_i)} m_l (x_i - x_l) + 2bm_i (x_i - x_d) \\
& + \frac{k(x_i - p_i)}{\sqrt{(x_i - p_i)^2 + (y_i - q_i)^2}} \\
= & 2b(\sum_{s_l \in S(s_i)} m_l + m_i) x_i - 2b(\sum_{s_l \in S(s_i)} m_l x_l + m_i x_d) \\
& + \frac{k(x_i - p_i)}{\sqrt{(x_i - p_i)^2 + (y_i - q_i)^2}} \\
= & 2b(\sum_{s_l \in S(s_i)} m_l + m_i) x_i - 2b(\sum_{s_l \in S(s_i)} m_l x_l + m_i x_d) \\
& - 2b(\sum_{s_l \in S(s_i)} m_l + m_i) p_i + 2b(\sum_{s_l \in S(s_i)} m_l + m_i) p_i \\
& + \frac{k(x_i - p_i)}{\sqrt{(x_i - p_i)^2 + (y_i - q_i)^2}} \\
\frac{\delta C_i}{dx_i} = & 2b(\sum_{s_l \in S(s_i)} m_l + m_i)(x_i - p_i) \\
& + 2b(\sum_{s_l \in S(s_i)} m_l + m_i) p_i - 2b(\sum_{s_l \in S(s_i)} m_l x_l + m_i x_d) \\
& + \frac{k(x_i - p_i)}{\sqrt{(x_i - p_i)^2 + (y_i - q_i)^2}}
\end{aligned}
$$

In addition to A, B_x, B_y, we define X_i and Y_i as follows:

$$X_i = x_i - p_i \qquad Y_i = y_i - q_i$$

so we can rewrite $\frac{\delta C_i}{dx_i}$ as

$$\frac{\delta C_i}{dx_i} = AX_i + B_x + \frac{kX_i}{\sqrt{X_i^2 + Y_i^2}}$$

Similarly, we compute $\frac{\delta C_i}{dy_i}$ and rewrite it as

$$\frac{\delta C_i}{dy_i} = AY_i + B_y + \frac{kY_i}{\sqrt{X_i^2 + Y_i^2}}$$

We first solve for the value for Y_i at which $\frac{\delta C_i}{dx_i} = 0$:

$$\frac{\delta C_i}{dx_i} = 0$$

$$AX_i + B_x + \frac{kX_i}{\sqrt{X_i^2 + Y_i^2}} = 0$$

$$\frac{kX_i}{\sqrt{X_i^2 + Y_i^2}} = -(AX_i + B_x)$$

$$\sqrt{X_i^2 + Y_i^2} = -\frac{kX_i}{AX_i + B_x} \tag{A.4}$$

$$X_i^2 + Y_i^2 = \frac{(kX_i)^2}{(AX_i + B_x)^2} \tag{A.5}$$

$$Y_i^2 = -X_i^2 + \frac{(kX_i)^2}{(AX_i + B_x)^2}$$

$$= \frac{X_i^2(k^2 - (AX_i + B_x)^2)}{(AX_i + B_x)^2}$$

$$Y_i = \pm\frac{X_i\sqrt{k^2 - (AX_i + B_x)^2}}{AX_i + B_x} \tag{A.6}$$

Since $k > 0$, (A.4) implies that

$$-\frac{X_i}{AX_i + B_x} > 0 \tag{A.7}$$

We now replace Y_i in $\frac{\delta C_i}{dy_i}$ by its value obtained above, and we solve for the value of X_i at which $\frac{\delta C_i}{dy_i} = 0$. We obtain:

$$\frac{\delta C_i}{dy_i} = 0$$

$$AY_i + B_y + \frac{kY}{\sqrt{X_i^2 + Y_i^2}} = 0$$

$$Y_i(A + \frac{k}{\sqrt{X_i^2 + Y_i^2}}) + B_y = 0$$

$$Y_i = -\frac{B_y}{A + \frac{k}{\sqrt{X_i^2+Y_i^2}}}$$

$$Y_i^2 = \frac{B_y^2}{(A + \frac{k}{\sqrt{X_i^2+Y_i^2}})}$$

$$\frac{X_i^2(k^2 - (AX_i + B_x)^2)}{(AX_i + B_x)^2} = \frac{B_y^2}{A^2 + \frac{k^2}{X_i^2+Y_i} + \frac{2Ak}{\sqrt{X_i^2+Y_i}}}$$

Using the value of $X_i^2 + Y_i^2$ obtained in (A.5), we get:

$$\frac{X_i^2(k^2 - (AX_i + B_x)^2)}{(AX_i + B_x)^2} = \frac{B_y^2}{A^2 + \frac{(AX_i+B_x)^2}{X_i^2} + \frac{2A}{|\frac{X_i}{AX_i+B_x}|}}$$

$$X_i^2(\frac{k^2}{(AX_i + B_x)^2} - 1) = \frac{B_y^2}{\frac{A^2X_i^2+(AX_i+B_x)^2}{X_i^2} + 2A|\frac{AX_i+B_x}{X_i}|}$$

(A.7) implies that

$$\left|\frac{X_i}{AX_i + B_x}\right| = -\frac{X_i}{AX_i + B_x}$$

$$X_i^2\left(\frac{k^2}{(AX_i + B_x)^2} - 1\right) = \frac{B_y^2}{\frac{A^2X_i^2 + (AX_i + B_x)^2}{X_i^2} - 2A\frac{AX_i + B_x}{X_i}}$$

$$X_i^2\left(\frac{k^2}{(AX_i + B_x)^2} - 1\right) = \frac{B_y^2}{\frac{A^2X_i^2 + (AX_i + B_x)^2}{X_i^2} - 2AX_i\frac{AX_i + B_x}{X_i^2}}$$

$$\frac{k^2}{(AX_i + B_x)^2} - 1 = \frac{B_y^2}{A^2X_i^2 + (AX_i + B_x)^2 - 2AX_i(AX_i + B_x)}$$

$$\frac{k^2}{(AX_i + B_x)^2} - 1 = \frac{B_y^2}{B_x^2}$$

$$(AX_i + B_x)^2 = \frac{k^2B_x^2}{B_x^2 + B_y^2}$$

$$(AX_i + B_x - \frac{kB_x}{\sqrt{B_x^2 + B_y^2}})(AX_i + B_x - \frac{kB_x}{\sqrt{B_x^2 + B_y^2}}) = 0$$

$$X_i = \frac{-B_x(\sqrt{B_x^2 + B_y^2} \pm k)}{A\sqrt{B_x^2 + B_y^2}}$$

and the corresponding Y_i is:

$$Y_i = \frac{-B_y(\sqrt{B_x^2 + B_y^2} \pm k)}{A\sqrt{B_x^2 + B_y^2}}$$

and hence

$$x_i = p_i + \frac{-B_x(\sqrt{B_x^2 + B_y^2} \pm k)}{A\sqrt{B_x^2 + B_y^2}}$$

$$y_i = q_i + \frac{-B_y(\sqrt{B_x^2 + B_y^2} \pm k)}{A\sqrt{B_x^2 + B_y^2}}$$

□

Theorem 5. *If no node can improve the current configuration U by moving by itself, then U is an optimal configuration*

Proof. The key intuition is that for a configuration in which no relay node can move and improve the cost by itself, the directional derivative in any direction at that configuration is positive; this is a sufficient condition for the optimality of that configuration because the cost function is convex.

We define the following notations. Let $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$ denote a final configuration where x_i and y_i represent the final positions of relay node s_i along the x-axis and the y-axis respectively. We note that x_i and y_i are fixed for the source nodes and the sink since these nodes do not move. The cost of this final configuration $c(x, y)$ can be represented as $T_x(x) + T_y(y) + M(x, y)$ where

$$\begin{aligned} T_x(x) &= b \sum_{ij \in E} m_i (x_i - x_j)^2 \\ T_y(y) &= b \sum_{ij \in E} m_i (y_i - y_j)^2 \\ M(x, y) &= k \sum_{i \in N} \sqrt{(x_i - p_i)^2 + (y_i - q_i)^2} \end{aligned} \tag{A.8}$$

Let $\alpha = (\alpha_1, \ldots, \alpha_n)$ and $\beta = (\beta_1, \ldots, \beta_n)$ denote the final configuration computed by our centralized algorithm. Let (g, h) be a possible direction in $\Re^2$: $g = (g_1, \ldots, g_n)$ represents the possible movement of each node along the x-axis and $h = (h_1, \ldots, h_n)$ the possible movement along the y-axis. We also define $G_i = (0, \ldots, 0, g_i, 0, \ldots, 0)$ and $H_i = (0, \ldots, 0, h_i, 0, \ldots, 0)$ to be the restriction of (g, h) where only node s_i moves. We want to show that the directional

derivative c' at (α, β) is positive in any direction (h, g). By definition,

$$c'((\alpha,\beta),(h,g)) = \lim_{t \downarrow 0} \frac{c(\alpha + tg, \beta + th) - c(\alpha, \beta)}{t}$$

By the properties of our algorithm, no node can reduce the global cost by moving by itself away from position (α, β). More specifically, if node s_i moves from position (α_i, β_i) to $(\alpha_i + tg_i, \beta_i + th_i)$, the global cost increases. Let Δ_i be this difference.

$$\begin{aligned}
\Delta_i \geq & 0 \qquad (A.9)\\
\Delta_i = & c(\alpha + tG_i, \beta + tH_i) - c(\alpha, \beta)\\
\Delta_i = & T_x(\alpha + tG_i) - T_x(\alpha)\\
& + T_y(\beta + tH_i) - T_y(\beta)\\
& + M(\alpha + tG_i, \beta + tH_i) - M(\alpha, \beta)
\end{aligned}$$

We replace $T_x, T_y,\ M$ in Δ_i and $c(\alpha + tg, \beta + th)$ by their values according to (A.8). We obtain the following:

$$\begin{aligned}
& c(\alpha + tg, \beta + th) - c(\alpha, \beta) =\\
& \quad \sum_{i=1}^{n} \Delta_i - 2b \sum_{i=1}^{n} m_i t^2 (g_i^2 + h_i^2)\\
& \quad + b \sum_{ij \in E} m_i t^2 (g_i - g_j)^2 + b \sum_{ij \in E} m_i t^2 (h_i - h_j)^2
\end{aligned}$$

We rewrite $-2b \sum_{i=1}^{n} m_i t^2 (g_i^2 + h_i^2) + b \sum_{ij \in E} m_i t^2 (g_i - g_j)^2 + b \sum_{ij \in E} m_i t^2 (h_i - h_j)^2$ in the previous

equation as t^2V for some value V independent of t. We get

$$\begin{aligned}\lim_{t\downarrow 0}\left(c(\alpha+tg,\beta+th)-c(\alpha,\beta)\right)/t &= \lim_{t\downarrow 0}\sum_{i=1}^{n}\Delta_i/t - tV \\ &= \lim_{t\downarrow 0}\sum_{i=1}^{n}\Delta_i/t \end{aligned} \tag{A.10}$$

We now examine the limit in (A.10) closely. By using values T_x, T_y and M again, we get:

$$\begin{aligned}\lim_{t\downarrow 0}\sum_{i=1}^{n}\Delta_i/t &= 2b\sum_{ij\in E} m_i(g_i-g_j)(\alpha_i-\alpha_j) \\ &+ 2b\sum_{ij\in E} m_i(h_i-h_j)(\beta_i-\beta_j) \\ &+ \lim_{t\downarrow 0}\Bigg(k\sum_{ij\in E}(\sqrt{(\alpha_i+tg_i-p_i)^2+(\beta_i+th_i-q_i)^2} \\ &-k\sum_{ij\in E}(\sqrt{(\alpha_i-p_i)^2+(\beta_i-q_i)^2}\Bigg)/t \end{aligned} \tag{A.11}$$

Using the difference of two squares identity, we can rewrite the limit term in (A.11) as

$$k\sum_{ij\in E}\frac{(\alpha_i-p_i)g_i+(\beta_i-q_i)h_i}{\sqrt{(\alpha_i-p_i)^2+(\beta_i-q_i)^2}} \tag{A.12}$$

(A.9), (A.11), (A.12) $\Rightarrow c'((\alpha,\beta),(h,g)) \geq 0$ so (α,β) is a global minimizer and represents an optimal configuration. □

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] http://www.k-team.com/robots/khepera/index.html.

[2] Cc1000 single chip very low power rf transceiver. http://focus.ti.com/lit/ds/symlink/cc1000.pdf.

[3] Cc2420 datasheet. http://inst.eecs.berkeley.edu/cs150/Documents/CC2420.pdf.

[4] S. Adlakha and M. Srivastava. Critical density thresholds for coverage in wireless sensor networks. In *Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE*, volume 3, pages 1615–1620, march 2003.

[5] S. Agarwal, R.H. Katz, S.V. Krishnamurthy, and S.K. Dao. Distributed power control in ad-hoc wireless networks. In *Personal, Indoor and Mobile Radio Communications, 2001 12th IEEE International Symposium on*, volume 2, 2001.

[6] D.M. Blough, M. Leoncini, G. Resta, and P. Santi. The k-neighbors approach to interference bounded and symmetric topology control in ad hoc networks. *Mobile Computing, IEEE Transactions on*, 5(9):1267–1282, 2006.

[7] Jonathan M. Borwein and Adrian S Lewis. *Convex Analysis and NonLinear Optimization. Theory and Exmaples*. Springer, 2000.

[8] Rainer E. Burkard. Selected topics on assignment problems. *Discrete Applied Mathematics*, pages 257–302, 2002.

[9] Jae-Hwan Chang and Leandros Tassiulas. Energy conserving routing in wireless ad-hoc networks. In *INFOCOM*, pages 22–31, 2000.

[10] Supriyo Chatterjea and Paul J. M. Havinga. An adaptive and autonomous sensor sampling frequency control scheme for energy-efficient data acquisition in wireless sensor networks. In *DCOSS*, pages 60–78, 2008.

[11] Jim Chou, Dragan Petrovic, and Kannan Ramchandran. A distributed and adaptive signal processing approach to reducing energy consumption in sensor networks. In *INFOCOM*, 2003.

[12] Karthik Dantu, Mohammad Rahimi, Hardik Shah, Sandeep Babel, Amit Dhariwal, and Gaurav S. Sukhatme. Robomote: enabling mobility in sensor networks. In *IPSN*, pages 404–409, 2005.

[13] Fatme El-Moukaddem, Eric Torng, and Guoliang Xing. Maximizing data gathering capacity of wireless sensor networks using mobile relays. In *IEEE MASS*, pages 312–321, 2010.

[14] Fatme El-Moukaddem, Eric Torng, Guoliang Xing, and Sandeep Kulkarni. Mobile relay configuration in data-intensive wireless sensor networks. In *IEEE MASS*, pages 80–89, 2009.

[15] S. R. Gandham, M. Dawande, R. Prakash, and S. Venkatesan. Energy efficient schemes for wireless sensor networks with multiple mobile base stations. In *Globecom*, 2003.

[16] Deepak Ganesan, Ben Greenstein, Denis Perelyubskiy, Deborah Estrin, and John Heidemann. An evaluation of multi-resolution storage for sensor networks. In *SenSys*, 2003.

[17] David Kiyoshi Goldenberg, Jie Lin, and A. Stephen Morse. Towards mobility as a network control primitive. In *MobiHoc*, pages 163–174, 2004.

[18] Y. Gu, D. Bozdag, and E. Ekici. Mobile element based differentiated message delivery in wireless sensor networks. In *WoWMoM*, pages 83–92, 2006.

[19] M.J. Handy, M. Haase, and D. Timmermann. Low energy adaptive clustering hierarchy with deterministic cluster-head selection. In *International Workshop on Mobile and Wireless Communications Network*, pages 368–372, 2002.

[20] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *HICSS*, 2000.

[21] John E. Hopcroft and Richard M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. Comput.*, 2(4):225–231, 1973.

[22] IEEE. Wireless medium access control (mac) and physical layer (phy) specifications for low-rate wireless personal area networks (lr-wpans). In *IEEE Standard 15.4*, 2003.

[23] Sushant Jain, Rahul Shah, Waylon Brunette, Gaetano Borriello, and Sumit Roy. Exploiting mobility for energy efficient data collection in wireless sensor networks. *MONET*, 11(3):327–339, 2006.

[24] David Jea, Arun A Somasundara, and Mani B Srivastava. Multiple controlled mobile elements (data mules) for data collection in sensor networks. In *DCOSS*, pages 244–257, 2005.

[25] Eun-Sun Jung and Nitin H. Vaidya. A power control mac protocol for ad hoc networks. In *Proceedings of the 8th annual international conference on Mobile computing and networking*, MobiCom '02, pages 36–47, 2002.

[26] Aman Kansal, David D. Jea, Deborah Estrin, and Mani B. Srivastava. Controllably mobile infrastructure for low energy embedded networks. *IEEE Transactions on Mobile Computing*, 5(8):958–973, 2006.

[27] Abtin Keshavarzian, Huang Lee, and Lakshmi Venkatraman. Wakeup scheduling in wireless sensor networks. In *MobiHoc*, pages 322–333, 2006.

[28] J.-H. Kim, D.-H. Kim, Y.-J. Kim, and K.-T. Seow. *Soccer Robotics*. Springer, 2004.

[29] Santosh Kumar, Ten-Hwang Lai, and József Balogh. On k-coverage in a mostly sleeping sensor network. In *MOBICOM*, pages 144–158, 2004.

[30] Euisin Lee, Soochang Park, Fucai Yu, and Sang-Ha Kim. On selection of energy-efficient data aggregation node in wireless sensor networks. *IEICE Transactions*, 93-B(9):2436–2439, 2010.

[31] Li Li and J.Y. Halpern. A minimum-energy path-preserving topology-control algorithm. *Wireless Communications, IEEE Transactions on*, 3(3):910–921, may 2004.

[32] N. Li, J.C. Hou, and L. Sha. Design and analysis of an mst-based topology control algorithm. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 3, pages 1702–1712, 2003.

[33] Yuhua Liu, Zhenrong Luo, Kaihua Xu, and Lilong Chen. A reliable clustering algorithm base on leach protocol in wireless mobile sensor networks. In *International Conference on Mechanical and Electrical Technology (ICMET)*, pages 692–696, 2010.

[34] Gang Lu, Narayanan Sadagopan, Bhaskar Krishnamachari, and Ashish Goel. Delay efficient sleep scheduling in wireless sensor networks. In *INFOCOM*, pages 2470–2481, 2005.

[35] Hong Luo, Jinge Wang, Yan Sun, Huadong Ma, and Xiang-Yang Li. Adaptive sampling and diversity reception in multi-hop wireless audio sensor networks. In *ICDCS*, pages 378–387, 2010.

[36] J. Luo and J-P. Hubaux. Joint mobility and routing for lifetime elongation in wireless sensor networks. In *INFOCOM*, pages 1735–1746, 2005.

[37] Liqian Luo, Qing Cao, Chengdu Huang, Tarek F. Abdelzaher, John A. Stankovic, and Michael Ward. Enviromic: Towards cooperative storage and retrieval in audio sensor networks. In *ICDCS*, page 34, 2007.

[38] Chia-Ching Ooi and Christian Schindelhauer. Minimal energy path planning for wireless robots. In *ROBOCOMM*, page 2, 2007.

[39] Vamsi Paruchuri, Shivakumar Basavaraju, Arjan Durresi, Rajgopal Kannan, and S. Sitharama Iyengar. Random asynchronous wakeup protocol for sensor networks. In *BROADNETS*, pages 710–717, 2004.

[40] Charles E. Perkins and Elizabeth M. Royer. Ad-hoc on-demand distance vector routing. In *WMCSA '99: Proceedings of the Second IEEE Workshop on Mobile Computer Systems and Applications*, page 90, Washington, DC, USA, 1999. IEEE Computer Society.

[41] R. Pon, M.A. Batalin, J. Gordon, A. Kansal, D. Liu, M. Rahimi, L. Shirachi, Y. Yu, M. Hansen, W.J. Kaiser, M. Srivastava, G. Sukhatme, and D. Estrin. Networked infomechanical systems: a mobile embedded networked sensor platform. In *Fourth International Symposium on Information Processing in Sensor Networks. IPSN*, pages 376–381, april 2005.

[42] Sylvia Ratnasamy, Brad Karp, Scott Shenker, Deborah Estrin, Ramesh Govindan, Li Yin, and Fang Yu. Data-centric storage in sensornets with ght, a geographic hash table. *MONET*, 8(4):427–442, 2003.

[43] Paolo Santi, Douglas M. Blough, and Feodor S. Vainstein. A probabilistic analysis for the range assignment problem in ad hoc networks. In *MobiHoc*, pages 212–220, 2001.

[44] Curt Schurgers, Vlasios Tsiatsis, and Mani B. Srivastava. Stem: Topology management for energy efficient sensor networks. In *IN IEEE AEROSPACE CONFERENCE*, pages 78–89, 2002.

[45] Mo Sha, Guoliang Xing, Gang Zhou, Shucheng Liu, and Xiaorui Wang. C-mac: Model-driven concurrent medium access control for wireless sensor networks. In *INFOCOM*, pages 1845–1853, 2009.

[46] R. Shah, S. Roy, S. Jain, and W. Brunette. Data mules: Modeling a three-tier architecture for sparse sensor networks. In *IEEE SNPA Workshop*, 2003.

[47] S. Shakkottai, R. Srikant, and N. Shroff. Unreliable sensor grids: coverage, connectivity and diameter. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 2, pages 1073–1083, march-3 april 2003.

[48] Yao Shen, Yunze Cai, Xinyan Li, and Xiaoming Xu. The restricted shortest-path-based topology control algorithm in wireless multihop networks. *Communications Letters, IEEE*, 11(12):937–939, december 2007.

[49] Suresh Singh, Mike Woo, and C. S. Raghavendra. Power-aware routing in mobile ad hoc networks. In *MOBICOM*, pages 181–190, 1998.

[50] Arun A. Somasundara, Aditya Ramamoorthy, and Mani B. Srivastava. Mobile element

scheduling with dynamic deadlines. *IEEE Transactions on Mobile Computing*, 6(4):395–410, 2007.

[51] Robert Szewczyk, Alan Mainwaring, Joseph Polastre, John Anderson, and David Culler. An analysis of a large scale habitat monitoring application. In *SenSys*, pages 214–226, 2004.

[52] Chiping Tang and Philip K. McKinley. Energy optimization under informed mobility. *IEEE Trans. Parallel Distrib. Syst.*, 17(9):947–962, 2006.

[53] Bülent Tavli and Onur Ceylan. Joint routing and multi level data compression for lifetime optimization in wireless sensor networks. In *ISCIS*, pages 692–696, 2009.

[54] J.-P. Tignol. *Galois' theory of algebraic equations*. John Wiley & Sons, Inc., New York, NY, USA, 1987.

[55] Yu-Chee Tseng, Chih-Shun Hsu, and Ten-Yueng Hsieh. Power-saving protocols for ieee 802.11-based multi-hop ad hoc networks. In *INFOCOM*, 2002.

[56] Tijs van Dam and Koen Langendoen. An adaptive energy-efficient mac protocol for wireless sensor networks. In *SenSys*, pages 171–180, 2003.

[57] P Varshney. *Distributed Detection and Data Fusion*. Springer-Verlag, New York, NY, 1996.

[58] Guiling Wang, Mary Jane Irwin, Piotr Berman, Haoying Fu, and Thomas F. La Porta. Optimizing sensor movement planning for energy efficiency. In *ISLPED*, pages 215–220, 2005.

[59] Lan Wang and Yang Xiao. A survey of energy-efficient scheduling mechanisms in sensor networks. *Mob. Netw. Appl.*, 11(5), 2006.

[60] Pu Wang, Rui Dai, and Ian F. Akyildiz. Collaborative data compression using clustered source coding for wireless multimedia sensor networks. In *INFOCOM*, pages 2106–2114, 2010.

[61] Qin Wang, Mark Hempstead, and Woodward Yang. A realistic power consumption model for wireless sensor network devices. In *IEEE SECON*, Reston, VA, September 2006.

[62] Wei Wang, Vikram Srinivasan, and Kee-Chaing Chua. Using mobile relays to prolong the lifetime of wireless sensor networks. In *MobiCom*, 2005.

[63] Z. Maria Wang, Stefano Basagni, Emanuel Melachrinoudis, and Chiara Petrioli. Exploiting sink mobility for maximizing sensor networks lifetime. In *HICSS*, 2005.

[64] Guoliang Xing, Tian Wang, Weijia Jia, and Minming Li. Rendezvous design algorithms for wireless sensor networks with a mobile base station. In *MobiHoc*, pages 231–240, 2008.

[65] Hongli Xu, Liusheng Huang, Yindong Zhang, He Huang, Shenglong Jiang, and Gang Liu. Energy-efficient cooperative data aggregation for wireless sensor networks. *J. Parallel Distrib. Comput.*, 70(9):953–961, 2010.

[66] Xue Yang and Nitin H. Vaidya. A wakeup scheme for sensor networks: Achieving balance between energy saving and end-to-end delay. In *IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 19–26, 2004.

[67] Ossama Younis and Sonia Fahmy. Distributed clustering in ad-hoc sensor networks: A hybrid, energy-efficient approach. In *INFOCOM*, 2004.

[68] Rong Zheng, Jennifer C. Hou, and Lui Sha. Asynchronous wakeup for ad hoc networks. In *MobiHoc*, pages 35–45, 2003.

[69] Daniel Zitterbart, Barbara Wienecke, James Butler, and Ben Fabry. Coordinated movements prevent jamming in an emperor penguin huddle. *PLoS one*, 6(6):e20260, 2011.