ENERGY-EFFICIENT SINK MOBILITY ALGORITHMS FOR WIRELESS SENSOR NETWORKS

A DISSERTATION SUBMITTED TO
THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
DOCTOR OF PHILOSOPHY
IN
COMPUTER ENGINEERING

By
Metin Koç
September, 2015

ENERGY-EFFICIENT SINK MOBILITY ALGORITHMS FOR WIRELESS SENSOR NETWORKS

By Metin Koç

September, 2015

We certify that we have read this dissertation and that in our opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Assoc. Prof. Dr. İbrahim Körpeoğlu(Advisor)

Prof. Dr. Özgür Ulusoy

Prof. Dr. Uğur Güdükbay

Prof. Dr. Adnan Yazıcı

Assist. Prof. Dr. Mehmet Köseoğlu

Approved for the Graduate School of Engineering and Science:

Prof. Dr. Levent Onural
Director of the Graduate School

ABSTRACT

ENERGY-EFFICIENT SINK MOBILITY ALGORITHMS FOR WIRELESS SENSOR NETWORKS

Metin Koç
Ph.D. in Computer Engineering
Advisor: Assoc. Prof. Dr. İbrahim Körpeoğlu
September, 2015

A wireless sensor network consists of a large number of tiny sensor nodes which are capable of sensing an environment and sending the collected data to a sink node. For most scenarios, sensor nodes are powered with irreplaceable batteries and this dramatically limits the lifetime of the network, especially due to overloading of the sensor nodes neighboring sink node. Such nodes need to forward more traffic than other nodes in the network. Moving sink node and in this way distributing forwarding-load evenly among sensor nodes is one of the important techniques for improving lifetime of sensor networks. We propose different mobility algorithms for single-sink and multiple-sink mobility problem to efficiently move sink nodes through a predefined set of sink sites.

We first provide packet-load and energy-load based sink mobility algorithms, called PLMA and ELMA, in which node-load parameters are incorporated into a table and this table is used to determine which sink site to visit in each round. We also give an integer programming model to get optimal results and do benchmarking. Since routing topology is an important component of sink mobility schemes, we also propose centralized and distributed routing topology construction algorithms to further increase network lifetime. Additionally, we propose an adaptive energy-load based sink movement algorithm, called A-ELMA, which does not require an initial training phase to learn about network topology. It incrementally constructs and updates energy-load table each time it visits a site location.

Finally, besides proposing algorithms for single-sink mobility problem, we also propose two different algorithms for multiple-sink mobility problem. Our Multiple Sink Movement Algorithm (MSMA) is a centralized algorithm and effectively limits the sink site combinations to reduce computation and communication overhead in scheduling sink movements without harming network lifetime significantly. Our Prevent and Move Away (PMA) algorithm is a fully distributed algorithm and does not require topology information to be collected. It selects sites based on

remaining energy values and distance metrics.

We evaluated our algorithms and compared them to some basic approaches in the literature by conducting extensive simulation experiments. Our simulation results show that our algorithms can perform better than some other alternatives in terms of network lifetime, latency and travel distance. We also identify under which conditions our algorithms perform better for each of these metrics. We observed that our algorithms provide simple-to-use, efficient, and effective solutions for single- and multiple-sink mobility problems in wireless sensor networks.

Keywords: wireless sensor network, energy-efficiency, sink-mobility, network lifetime improvement, routing tree construction.

ÖZET

KABLOSUZ ALGILAYICI AĞLAR İÇİN ENERJİ VERİMLİ ALICI HAREKETLİLİK ALGORİTMALARI

Metin Koç
Bilgisayar Mühendisliği, Doktora
Tez Danışmanı: Doç. Dr. İbrahim Körpeoğlu
Eylül, 2015

Kablosuz algılayıcı ağ, bir ortamı algılayabilen ve toplanan veriyi alıcı düğüme ileten çok sayıda küçük algılayıcı düğümden oluşur. Çogu senaryoda algılayıcı düğümler değistirilemez pillerle çalıstırılırlar ve bu durum alıcı düğüme komşu algılayıcı düğümlerin aşırı yüklenmesine bağlı olarak ağ yaşam süresini önemli ölçüde sınırlar. Alıcı düğümü hareket ettirme ve bu yolla algılayıcı düğümler arasındaki iletme yükünü eşit olarak dağıtmak, algılayıcı ağların yaşam süresini iyileştirmek için önemli tekniklerden biridir. Alıcı düğümleri, önceden tanımlanmış bir dizi alıcı yerine doğru etkili biçimde hareket ettirmek üzere tek ve çoklu alıcı hareketlilik problemleri için farklı hareketlilik algoritmaları öneriyoruz.

Öncelikle, PLMA ve ELMA adı verilen düğüm yükü parametrelerinin bir tabloda birleştirildiği ve bu tablonun her bir tur için hangi alıcı yerine gidileceğini belirlemek için kullanıldığı, paket ve enerji yükü bazlı alıcı hareketlilik algoritmaları öneriyoruz. Ayrıca en iyi sonuçları elde etmek ve karşılaştırmalı değerlendirme yapmak amacıyla bir tamsayı doğrusal programlama modeli de veriyoruz. Yönlendirme topolojisi alıcı hareketlilik planlarının önemli bir bileşeni olduğundan, ağ yaşam süresini daha da artıran merkezi ve dağıtık yönlendirme topoloji oluşturma algoritmaları da öneriyoruz. İlaveten, ağ topolojisini öğrenmek için başlangıçta öğrenme aşamasına gerek duymayan A-ELMA adlı enerji-yük bazlı alıcı hareketlilik algoritması öneriyoruz. Bu algoritma bir yeri her ziyaret ettiğinde enerji yük tablosunu aşamalı olarak oluşturup güncellemektedir.

Son olarak, tek alıcı hareketlilik problemi için önerilen algoritmalar dışında çoklu alıcı hareketlilik problemi için de iki farklı algoritma öneriyoruz. MSMA adlı çoklu alıcı hareket algoritması ağ yasam süresine önemli ölçüde zarar vermeyecek şekilde alıcı hareketlerinin zamanını programlarken hesaplama ve iletişim ek yükünü azaltmak için alıcı yer kombinasyonlarını etkili biçimde sınırlayan merkezi bir algoritmadır. PMA algoritması ise topoloji bilgisinin toplanmasını gerektirmeyen tamamen dağıtık bir algoritmadır. Bu algoritma, yerleri, kalan

enerji değeri ve uzaklık metriklerine göre seçmektedir.

Algoritmalarımızı değerlendirmek ve literatürdeki bazı temel yaklaşımlarla karşılaştırmak için kapsamlı benzetim deneyleri gerçekleştirdik. Benzetim sonuçları algoritmalarımızın ağ yaşam süresi, gecikme ve katedilen mesafe açısından bazı diğer seçeneklerden daha iyi sonuç verdiğini göstermektedir. Ayrıca algoritmalarımızın bu metriklerin her biri için hangi koşullarda daha iyi sonuç verdiğini belirliyoruz. Algoritmalarımızın kablosuz algılayıcı ağlardaki tek ve çoklu alıcı hareketlilik problemi için kullanımı kolay, etkin ve verimli çözümler sağladığını gözlemledik.

Anahtar sözcükler: kablosuz algılayıcı ağ, enerji verimliliği, alıcı hareketliliği, ağ yaşam süresi geliştirme, yönlendirme ağaç yapılandırması.

Acknowledgement

First of all, I would like to express my sincere gratitudes to my advisor Assoc. Prof. Dr. İbrahim Körpeoğlu. Dr. Körpeoğlu not only guided me to find my way in this long academic journey, but also taught me how to be strong against difficulties in life. I will be grateful for the things he has patiently taught me.

I would like to thank members of the thesis progress committee, Prof. Dr. Özgür Ulusoy and Prof. Dr. Adnan Yazıcı for their valuable comments and discussions for the last four years. I would also like to thank Prof. Dr. Uğur Güdükbay and Assist. Prof. Dr. Mehmet Köseoğlu for being on my thesis defense jury and their useful feedback.

I have had several people around during this long journey who made me feel special. I definitely learnt a lot from the discussions we had during lunchtime with my dear friends Özgür Şanlı and Özgür Ergül. Levent, my brother, has always been there when I needed to talk despite the physical distance and time difference. Finally, I am grateful to my parents for their continuous love and support which kept me walking in life.

Lastly, I can't emphasize enough how grateful I am to my beloved wife, Buket, who has an incredible support to put a happy ending to this story. She encouraged me every single time I felt desperate during these years. I apologize from my children, Deniz and Ali Aras, for all this time that we couldn't spend together. I'm hoping that we will make up this time together from now on.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A wireless sensor network (WSN) is generally composed of numerous tiny sensor nodes and a more powerful special node called a sink node [1]. Sensor nodes are low-cost and low-power devices that are composed into a network to sense the environment, process the data, and send the information to the sink node. Each sensor node typically has a micro-controller (processing data, controlling other components), one or more sensors (sensing the environment), a memory chip (storing program and data), a transceiver (sending and receiving data), and a power source (mostly batteries).

Network-enabled sensor nodes are being produced starting from 2000s and the technology is constantly evolving. To give an example, some industrial sensor nodes have 512 KB program memory and 16 MB data memory on board [2]. These nodes support 802.15.4/ZigBee or Bluetooth for data communication. Although built-in sensors on these nodes differ according to the application, they can typically be temperature, humidity, light and acoustic sensors.

Sinks, or base stations, have similar but more powerful units on them. Sinks are generally attached to a computer and are mains-powered, therefore do not have energy problem. A sink node acts as a gateway between sensor nodes and an application (end user). There can be either one or multiple sinks in the

environment depending on the application WSNs are used for.

Sensor networks have various applications. Although different taxonomies are done for these applications, an important category is monitoring and tracking applications [3,4]. Sensor networks applications in the sectors of military, environment, healthcare and industrial automation have both monitoring and tracking features [5]. Military applications include border surveillance, monitoring friendly forces, or tracking enemy soldiers. Environment applications include detecting forest fires, detecting chemical material leakage, and animal tracking. Industrial applications include guided vehicle tracking and manufacturing process monitoring. Healthcare applications mainly include patient monitoring and tracking. Using sensor nodes for healthcare applications is one of the important emerging markets for WSNs [3].

Sensor nodes have different power sources, like batteries, micro-fuel cells, radioactive power sources, and micro heat engines [6]. Energy harvesting is another possible power source and becomes popular in recent years [7–9]. However, in most cases, batteries are the main source of power supply. Most of the sensor nodes in the industry use AA batteries (3.0 Volt) for power source. These batteries have very limited capacity and need to be changed or recharged regularly to have an extended lifetime. However, in typical sensor network applications, like habitat monitoring, enemy tracking, or border surveillance, it is very hard to access the nodes and change their batteries when needed.

This limited energy problem is the major bottleneck for sensor networks which should be carefully considered when designing sensor network protocols, algorithms, or applications to prolong network lifetime. Different energy-efficient approaches in various network layers are proposed in the literature for improving network lifetime, such as power control mechanisms (physical layer) [10,11], energy–efficient MAC layer protocols (data-link layer) [12,13], and data-gathering/routing protocols (network layer) [14,15].

Sink mobility is another approach for improving network lifetime. When there

is no data aggregation applied in the network, sensor nodes transmit their messages and relay the data packets of other nodes. This situation causes the nodes in the vicinity of the sink to deplete their energy faster than other nodes in the network, which is called the *hot-spot* problem [16]. Making the sink node mobile can help solving this problem via distributing the hot-spot load among all sensor nodes, in this way improving the network lifetime.

Sink mobility can be either *uncontrolled* or *controlled*, depending on the mobility scheme used [17]. In uncontrolled mobility, data MULEs (Mobile Ubiquitous LAN Extensions) are used as a middleware between sinks and sensor nodes [18]. These MULEs collect data from sensor nodes when they come within range, then they buffer the data and send it to the sink [19]. This type of mobility is called uncontrolled, because MULEs do not consider current network conditions, but instead move randomly in the area. In controlled mobility, however, node and network conditions like remaining energy or node density are taken into consideration.

Until now, controlled sink mobility problem is generally handled as an optimization problem and mathematical programming based solutions are given. Although it is important to see upper bounds on the network lifetime, most of these solutions do not scale and also cannot be run directly on the sensor nodes due to their high computation and space requirements. Therefore, it is important to give algorithms that can be run on the sensor devices directly, instead of doing calculations on powerful centralized machines and then configuring the network.

In this thesis, our aim is to prolong the lifetime of sensor nodes and network using controlled sink mobility algorithms. We investigate both single-sink and multi-sink problems and propose centralized and distributed algorithms to coordinate sink movements. We also give energy-efficient tree-topology construction algorithms. Routing tree construction is an important component of protocols for sensor networks with a mobile sink, since the routing topology needs to be repeatedly re-constructed every time the sink changes its location.

Firstly, we investigate the single-sink mobility problem. We present two different centralized algorithms, Energy-Load based Movement Algorithm (ELMA) and Packet-Load based Movement Algorithm (PLMA), to arrange sink movements that include determining the order of visits and the number of rounds to stay in a migration point. Knowing the number of packets each node sends to its parent or the amount of energy each node consumes while transmitting packets in a round for each migration point can help the sink node to select next migration points in a way to balance the energy consumption in the network. This information can be obtained in two ways: by using node locations directly or by visiting each alternative site before network starts its operation. Energy-efficient tree topologies can be constructed since nodes' neighbor relationships and logical levels can be extracted from location information for a specific sink site. However, this cannot be possible for most of the cases. So in our case, a sink node visits possible sites before network starts its operation and forms tree-based topologies to calculate the number of packets and energy expenditure values. Using these values the sink constructs traffic- or energy-load matrices. Our algorithms use these matrices together with node remaining energy values to select next migration points.

We also formulate the single-sink mobility problem as an integer program to obtain the optimal results. These results provide a benchmark for us to evaluate the performance of our proposed ELMA and PLMA algorithms. We performed extensive simulations to compare the performance of ELMA and PLMA with optimal, random and static sink approaches. Results showed that ELMA can perform up to five times more lifetime than static case and two times than randomly selecting the sink sites approach while it is just five percent below the optimal solution. ELMA and PLMA run faster than optimal algorithm, especially when number of nodes increases. The memory that is needed to store these matrices can easily be supported with current sensor node technology.

Although ELMA and PLMA improve network lifetime, it is assumed that the same tree-topology is formed, i.e., each node selects the same parent, each time the sink visits the same location. Mechanical problems in sensor nodes or some obstacles in the field like an animal in a forest can prevent nodes to select the same

parent when sink visits the same site. Besides that, a training phase is required for sink to visit each site and construct tree-based topologies to populate traffic- and energy-load matrices. This phase delays the network to start its operation and can be quite long depending on the speed of the sink node. Such a delay cannot be tolerated in applications like forest fire detection or border surveillance.

We propose an adaptive version of our ELMA algorithm, called A-ELMA, which does not require any network information a priori. A sink visits sites mostly randomly in the first rounds of the network operation, constructs topologies and learns traffic/energy loads of nodes. It also updates those values when visiting the same site again and does not restrict the nodes to select the same parent again and again. Instead, it takes advantage of constructing energy-efficient tree-topologies considering nodes' residual energy values. Experimental results showed that A-ELMA performs better than ELMA and optimal algorithms.

When sink is mobile in a network, it changes its position frequently and reconstructs a topology each time it visits a site location. Constructing energy-efficient tree topologies can prolong the lifetime jointly with mobility [20]. Therefore, energy-efficient topology construction is a complementary component of mobility scheme and increases the effectiveness of mobility to improve network lifetime. We propose two algorithms, a centralized one (CTCA) and a distributed one (DTCA), to construct an energy-efficient routing topology each time sink changes its location.

In our Centralized Topology Construction Algorithm (CTCA), nodes send their parent list and level information to the sink node where balanced topology is constructed considering packet load and remaining energy values of the nodes. In our Distributed Topology Construction (DTCA) algorithm, however, each node selects a parent based on probabilities proportional with remaining energy values. The idea is to balance the load of the sink's first-hop neighbors as much as possible without direct communication with neighbor nodes. Our results show that CTCA improves lifetime better than DTCA, however, CTCA algorithm incurs more delay during construction since all nodes spend time in sending their values to the sink node and wait for the configuration information to come back.

The controlled single-sink mobility problem is widely investigated in the literature [20–26]. Mathematical programming formulations that can maximize the lifetime under various constraints are used in most works. However, the multiple-sink mobility problem has specific challenges compared to its single-sink counterpart, especially during the site selection process. In the single-sink mobility problem, there is no competition for sink sites. The sink chooses the most energy-efficient site among the alternatives and moves to it without considering whether it has been occupied already by another sink.

In the multiple-sink mobility problem, s sinks should choose different sites among p alternatives in a mutually exclusive manner so that any two sinks do not decide to move to the same location at the same re-configuration instance. In the single-sink case, the base station chooses one point out of p alternatives, but in the multiple-sink case, the number of possible combinations to select in each decision is $\binom{p}{s}$. These make the multiple-sink problem more complex compared to the single-sink problem.

Although they should not be used in a real sensor node environment due to computational constraints, mathematical-programming based solutions are mostly preferred, as in single-sink case, for maximizing network lifetime using multiple-sink movements. Additionally, an exponential number of constraints can exist in these formulations, since all possible sink-site combinations are included. Solving these formulations can take several days even with current computation technology. These calculations must be done before the network begins operation and the solution, sink sojourn time for each sink site, should be given to the sinks. Obviously, these solutions are not dynamic and fail to adapt to changes, such as unavailable sites or node failures due to physical conditions, that can occur during network lifetime.

We propose Multiple-Sink Movement Algorithm (MSMA) and Prevent and Move Away (PMA) Algorithm for multiple-sink mobility problem. In MSMA, which is a centralized algorithm, we have a threshold value which determines how much load-related information (packet load or energy load) each sink can keep for scheduling their movements. Load-related information indicates the

energy consumed at a node or the traffic-amount a node should handle for a particular sink site combination for multiple sinks. We select a subset of sink-site combinations using the k-means algorithm, which does not exceed the threshold value. Sinks select sites randomly with a small probability to increase the number of combinations, again respecting the threshold value.

PMA is a distributed algorithm, on the other hand, in which each sink evaluates the information coming from the nodes on its subtree and forbids some of the sites that are close to its descendants. Each sink selects its candidate sites that have nodes with more energy and shares this information with the other sinks to select $s/2$ sites that have global maximum value. The remaining $s/2$ sites are selected farthest from these sites to make a balance between energy and distance without global network parameter values. Simulation results show that MSMA performs better than MinDiff-RE [27], random movement and static sink approaches. Although PMA's performance is worse than MSMA, it performs better than random movement and sinks travel less compared to other algorithms when PMA is used.

1.1 Contributions

Our contributions in this dissertation can be summarized as follows:

- We proposed two centralized single-sink mobility algorithms which consider nodes' energy or packet load values to schedule sink movements for prolonging network lifetime.
- We also formulated the single-sink mobility problem as an integer program to determine the number of rounds sink should stay at each point to maximize the lifetime. We used the results of this mathematical model as a benchmark for evaluating the results of our traffic- and energy-load based sink mobility algorithms.
- To evaluate the performance of the algorithms, we developed a simulation

environment in MATLAB and integrated IBM Cplex Optimizer to solve the problem in the same environment. We showed via simulations that our energy-load based algorithm is just five percent below the optimal solution while performing up-to five times better than static sink case and two times better than random movement approach.

- We proposed two energy efficient routing tree construction algorithms, CTCA and DTCA. CTCA uses network information to carefully arrange the loads of nodes, while in DTCA each node selects a parent in a probabilistic manner without communicating with its neighbors.

- We proposed an adaptive single-sink mobility algorithm (A-ELMA) that removes the assumptions used in our energy-load based algorithm. Our adaptive algorithm does not need to learn network parameters via a training phase. Instead it learns these parameters while visiting the sites and updates them to take advantage of residual energy of sensor nodes.

- We proposed centralized and distributed mobility algorithms to coordinate the movement of multiple sinks. Our centralized MSMA algorithm limits the possible number of site combinations according to a threshold value to reduce complexity. Our PMA algorithm is fully distributed and does not require any prior network information.

1.2 Outline of the Dissertation

The rest of the dissertation is organized as follows. In the next chapter we give the related work about single-sink mobility, tree-topology construction and multiple-sink mobility problems. In Chapter 3, we give traffic- and energy-load based algorithms for single-sink mobility problem. Chapter 4 describes our energy efficient tree-topology construction algorithms and our adaptive sink mobility algorithm. We present our algorithms for multiple-sinks mobility problem in Chapter 5. Finally, we give our conclusions and future research directions in Chapter 6.

Chapter 2

Related Work

In this chapter we give a summary of the previous work related to our research problem. First, we give studies about single-sink mobility problem. Next, we discuss the energy-efficient data gathering schemes, logical topologies and constructing tree-topologies without data aggregation. Lastly, we summarize some of the previous work about multiple-sink mobility problem.

2.1 Single-Sink Mobility

The most important characteristic of WSNs is the limited energy resources of sensor nodes. A typical sensor node has generally an irreplaceable limited-capacity battery. Consuming the least amount of energy is the most critical criterion while designing any sensor-network protocol so that the energy of nodes and network is utilized as efficiently as possible.

Several approaches are proposed in literature to minimize a sensor network's energy consumption and thus to improve network lifetime: adjusting the transmit power depending on the distance to the receiver when sending messages [28], developing energy-efficient MAC or routing protocols [29–34], minimizing the number of messages traveling in the network, and putting some sensor nodes

into sleep mode and using only a necessary set of sensor nodes for sensing and communication [35].

Moving the sink node in the deployment region is another approach to prolong network lifetime, because this eliminates the hot-spot problem around the sink node [25,26].

[36] examines the sink mobility problem with multiple base stations. The main motivation behind this choice is to have more options for routing, thus reducing the hop-count to prolong network lifetime. The study presents two different integer linear programming (ILP) formulations to relocate the sinks (there are three sinks), one which has the objective function to minimize the maximum energy spent by a sensor node, and the other which has the objective function to minimize the total energy consumption in a round, subject to some constraints. The authors also examine the impact of the number of available base stations over the network lifetime and find that increasing the number of base stations beyond a certain threshold does not improve lifetime. This is because when the number of base stations is increased to a sufficient number, each sensor node can transmit messages to one of the base stations by direct communication.

Mobility algorithms and routing are considered together in [20]. It is assumed that sensors are densely deployed within a circle with a Poisson distribution. The authors define the network lifetime as the time until the *loss of coverage* starts. They specify a linear programming (LP) formulation to minimize the load on each one of N sensor nodes. In their solution, they first find the optimal mobility strategy by fixing the routing strategy as shortest path, then using the resulting strategy they try to find a final routing scheme with better performance than shortest path. The authors prove that the optimal mobility strategy is the trajectory around the periphery of the network. They find a better routing strategy by concentrating on an inner circle in the network area and develop a heuristic based on this. However, since they do not compare their results to any other mobility approach, we are unable to judge the relative performance of their scheme. The main drawback of their work is the assumption that the network region is circular. There is no explanation of how the solution may be adapted

to other region types.

A work more similar to ours is presented in [21]. In this paper, N sensor nodes and a sink node are randomly deployed to an area of interest. It is assumed that each sensor node generates data with a constant rate and there is a certain set of locations where the sink can move. The authors present two complementary algorithms for solving the sink mobility and routing problem. One is a *scheduling* algorithm that determines the duration for each candidate sink site where the base station can stay, and the other is a *routing* algorithm that determines the most energy-efficient path from each sensor node to the sink. An LP formulation is given which maximizes the sum of sink sojourn times at all possible locations (subject to some constraints) and mobile and static sink approaches are compared with different routing schemes. The simulations use two scenarios, including just four and five sink sites. These are quite limited scenarios. The authors also perform experiments by adding a routing parameter.

In [37], the authors give a routing algorithm, *mobiroute*, to route data towards a mobile sink for improving network lifetime. Mobiroute extends MintRoute proposed by [38] by adding functions to perform some operations like notifying a node for link failures, informing all nodes about topology changes, and minimizing the packet loss during mobility. They propose a two-phase adaptive algorithm to control sink mobility. In *initialization* phase, mobile sink builds a power consumption profile for each anchor point and drops some of them if their weights are low. In *operation* phase, the mobile sink stays at chosen points and updates profiles. Simulation results show that making the sink mobile improves network lifetime without sacrificing reliability in packet delivery.

The work of [22] is a detailed study about controlled sink mobility. The authors present a centralized mixed integer linear programming (MILP) model that determines sojourn times and the order of visits to sink sites. Moreover, they develop a distributed and localized heuristic called *greedy maximum residual energy* (GMRE), as a solution to the problem. The network model is similar to the one given in [21], but unlike that model, the deployment area is divided into grids and the corners of these grids are determined as sink sites. The authors introduce

two parameters to make the model more realistic: a d_{max} parameter represents the upper bound for the distance the sink can travel between the current and the next site, and a t_{min} parameter defines a mandatory time the sink is required to stay at a site. The authors evaluate the performance of MILP, GMRE, random movement (RM), and static sink (STS) schemes. The first two give better results than the others, with MILP performing 30% to 50% better than GMRE for increasing t_{min} values.

Optimal sensor deployment, scheduling, routing and sink mobility are all considered together for maximizing lifetime in [39]. The authors give a mixed integer linear programming model, which considers the aforementioned parameters together to maximize network lifetime. However, the authors present also two heuristics, since current state-of-the-art MILP solvers fail to solve problems with realistic size in an acceptable time frame. The *period iteration* heuristic (PIH) limits the number of total periods and increments it one by one up until no further improvement is achieved. The *sequential assignment* heuristic (SAH) solves three different subproblems for determining the values of sensor locations, activity schedules, and routing decision variables.

Although centralized algorithms are proposed in previous works, [40] presents a distributed algorithm for delay-tolerant wireless sensor networks. The authors aim to maximize the number of tours (T), where each tour takes D (maximum delay tolerance) time units. They propose *delay tolerant mobile sink model* (DT-MSM) and decompose it into three subproblems. These algorithms are then combined to a main algorithm and convergence analysis is done. Experimental results are presented to verify the validity of the algorithm.

Previous works that use (M)ILP force the authors to limit the number of sink sites and the number of nodes in their simulations. For example, [36] uses maximum 30 nodes while [21] uses 100 nodes. Since solving IP or MILP is NP complete [41], increasing the number of nodes will cause a dramatic increase in deciding sink movements. WSNs, however, are likely to contain tens of thousands of nodes, especially as the cost of sensor nodes becomes cheaper. Additionally, since sensor nodes do not have large buffer capacities, most nodes will not be

able to tolerate long latencies, which will cause a large packet loss rate. In such cases, heuristic algorithms are more efficient and more effective than an optimal solution. Therefore, in this thesis, we focus and provide heuristic algorithms that can be used during network operation. Our algorithms maintain a node-load table and give decisions based on such a table instead of performing long and static pre-calculations for optimal results.

2.2 Tree-based Topology Construction

Data gathering/collection is an important task in wireless sensor networks, since nodes are mostly deployed to a region of interest to learn about some environmental parameters like temperature or humidity [42]. Several logical topology types are proposed in literature to collect information from sensor nodes to the sink. [43] classifies these topologies into five categories: flat/unstructured topologies, chain-based topologies, cluster-based topologies, tree-based topologies, and cluster-tree topologies.

Flooding or gossiping is used in a flat topology to establish paths on demand to forward data packets toward the sink. SPIN [44] and Directed Diffusion [45] are well-known protocols that use flat/unstructured topologies. A chain is constructed and a node is selected as the leader in a chain-based topology. Other nodes communicate with each other and with the sink using this chain. PEGASIS [46] is an example that uses chain-based topology. A hierarchical structure is constructed in a cluster-based topology (CBT). Cluster heads (CH) are elected and clusters are formed in the region. Sensor nodes become cluster members and send their data to CHs to be forwarded to sink. LEACH [47] and HEED [48] are important algorithms utilizing cluster-based topologies. In a tree-based topology, which is also used in our work, nodes select parents and construct a logical tree topology rooted at the sink node. Finally, cluster-tree topology (CTT) is a hybrid type, which combines clustering and tree formation. A designated device (DD), a special node with higher transmission power, is selected to act as a cluster head. Nodes are added to DD to form a cluster-tree using the beacons they receive

from the neighbors. Cluster-Tree Data Gathering Algorithm (CTDGA) [49] is a solution based on cluster-tree topology.

Tree-based topologies have pros and cons compared to other alternatives. Tree-based topologies consume less power than flat topologies and save more energy compared to cluster-based topologies via avoiding packet flooding [43,50,51]. However, tree formation is costly and is not resilient to node failures. Therefore, tree maintenance has high overhead when sink changes locations.

Data aggregation, combining data and applying some aggregation functions like sum or average, is an important design parameter when designing a tree topology formation algorithm. Usage of data aggregation depends on application scenario. It can be applied when average temperature of a region is queried, for example. On the other hand, data aggregation cannot be considered when collecting images from different regions of a battlefield [52]. When data aggregation is used in a WSN, each node sends one packet to its parent at each round, regardless of its number of descendants. When aggregation is not used, however, nodes closer to the sink node suffer from unbalanced packet load, since they not only send their packets but also forward the packets of their descendants.

Lifetime improvement with data gathering and aggregation is widely investigated in the literature [53–62]. In this thesis, we assume that data aggregation is not used and therefore we provide solutions for no-aggregation scenarios. We will not go into details of works that assume data aggregation, however, [63] is notable here, since the proposed algorithm gives near optimal results in constructing maximum-lifetime data gathering trees.

Constructing tree topologies without data aggregation is not investigated as much as the ones with data aggregation. [52] gives a $\Omega(log\, n/log\, log\, n)$ approximation algorithm to solve this problem. The authors formalize the problem as finding a maximum lifetime tree (*min-max-weight spanning tree*) among multiple trees to maximize the network lifetime via balancing the load in bottleneck nodes as much as possible according to their energy levels. This problem is similar to capacitated spanning tree problem, i.e., finding a minimum weight spanning tree

in which every subtree contains fewer than K vertices, which is an NP-complete problem. The authors give MITT algorithm, which is near-optimal, to solve constructing maximum lifetime tree for data gathering without aggregation problem. The algorithm starts with an arbitrary tree by using breadth first traversal on the network. The *capacity* of each node in its ability in serving number of descendants proportional with its remaining energy value is calculated. MITT algorithm sorts the nodes with respect to their weights in decreasing order and the weights of the nodes with lower capacities are transferred to the nodes with higher capacities. Experimental results show that MITT provides better network lifetime compared to other data gathering algorithms, such as PEDAP, MNL, etc. However, its running time is much longer than those algorithms, especially when number of nodes increases.

An exact algorithm for maximum lifetime data gathering tree is given in [64]. First, the authors prove that the problem cannot be approximated within a factor greater than 2/3. They decompose the problem into two subproblems to reduce the search space when scanning all spanning trees to find an optimal spanning tree. The algorithm has three parts. In the first part, a partial spanning tree is initialized. Graph is decomposed into several blocks, each block is solved and solutions are merged in the second part. In the last part, edges are tried to be included into the tree. The algorithm has $O(nm)$ space complexity, where n is the number of nodes and m is the number of edges. However, the algorithm runs in exponential time in the worst case. Simulation results show that MITT algorithm [52] runs faster than the exact algorithm and gives results close to the optimal in most of the cases. However, MITT's worst case is poor, since it can give results less than half of the optimal in some cases. The running time of the exact algorithm is considerably faster, taking a few seconds, than enumerating all spanning trees, which may take several hours.

[65] gives another centralized algorithm for constructing maximum-lifetime data gathering tree. The authors classify the nodes with respect to their packet loads: high, medium, and low. MLTTA algorithm starts with an initial tree. Each node level and initial parent are selected during this step. The class of each node, high, medium or low, is determined using load calculation formulas.

The descendants of high-load nodes are transferred to low-load nodes as much as possible to balance the load among the nodes. This transfer operation is repeated until each of the high-load node's descendants are assigned to low-load nodes. The authors compare the performance of MLTTA algorithm with breadth first traversal (INITA) and randomly generated tree (NAIVE). Experimental results show that it improves the lifetime compared to INITA and NAIVE when sink is at the center, but the gap is closed when sink is located at the corners. Although [65] is not comparing its algorithm's performance with MITT algorithm, results in [66] show that the lifetime achieved using MLTTA algorithm is about one third of MITT's performance.

Aforementioned algorithms that are given for constructing energy efficient data gathering trees are all centralized. Weighted Spanning Tree Distributed Optimization (WSTDO) algorithm, which is a distributed algorithm performed locally on the spanning trees, is given in [66]. Authors first categorize the nodes according to their load or energy levels. Red nodes are bottleneck nodes and have large load or low energy. Grey nodes can be bottleneck nodes if they would accept more load. Green nodes have low load or high energy. Weights can be transferred from red nodes to green nodes. WSTDO consists of four phases: initialization, tree maintenance, tree optimization, and data transfer. Initial tree is constructed in initialization part. Current topology information is collected and possible cycles are detected and recycled in tree maintenance phase. Nodes' loads are locally optimized by using capacity calculations. Finally, data is transferred over the constructed tree in previous phases. WSTDO performs worse than MITT algorithm. Its performance is almost same when compared to MLTTA. However, it has less overhead, especially for sparse networks, and handles node and link failures locally.

2.3 Multiple-Sink Mobility

Although it has not been investigated as much as its single-sink counterpart, there are some studies about the multiple-sink mobility problem.

[67] presents one of the earliest works about the multiple-sink mobility problem. The authors divide the sensor network lifetime into equal periods of time, called rounds, and relocate the sinks at the start of each round. They present an integer linear program (ILP) that minimizes the maximum energy spent in each round in order to determine the locations of the sinks. A variation of this model minimizes the total energy consumption in a round. They compare these two ILP formulations as well as the random and static-sink approaches. Minimizing the maximum energy spent increases network lifetime significantly compared to other approaches.

[27] also deals with the multiple-sink mobility problem. The authors choose feasible sink-site locations along the periphery of the field and propose three heuristics, Top-K_{max}, Max-Min-RE, and MinDiff-RE, to determine the sojourn points of the sinks in each round. In Top-K_{max}, K sites are chosen, in which the nearest neighbor nodes have maximum residual energies. The Max-Min-RE heuristic aims to maximize the minimum residual energy for all combinations of sink sites and sinks for a given routing algorithm. The MinDiff-RE method uses the same approach as Max-Min-RE, except the goal is to minimize the difference in residual energy. Experiments show that MinDiff-RE gives better network lifetime compared to other approaches. Although these heuristics are presented as energy-efficient and low-complexity algorithms, Max-Min-RE and MinDiff-RE need to process all combinations before selecting the next migration points. These methods can be practical for small values of number of sinks and sites. However, in typical scenarios, for instance for 30 sink sites and 8 sink nodes, there may be millions of sink-site combinations to consider, which makes these solutions impractical.

[68] proposes a linear programming (LP) model as well as centralized and heuristic algorithms for multiple-sink mobility problem. Any combination of s sinks occupying s locations among k sites is called a *configuration*. The authors define an LP model with the goal to find a sequence of configurations that maximizes network lifetime. Although the LP has exponential number of constraints, the authors use a separation algorithm to resolve the LP a polynomial number of times. They also give a centralized heuristic that uses the output of the LP and

runs in polynomial time. Finally, authors define a deployable distributed heuristic for coordinating the motion of multiple sinks throughout the network. The simulation results show that the proposed schemes improve lifetime significantly compared to static-sink case and random mobility.

[69] formulates the network lifetime maximization for the h-hop constrained multiple-sink mobility problem such that the total travel distance of each sink is bounded by L and the maximum number of hops from each sensor to a sink is bounded by h, where $h \geq 1$. The authors show that the problem is NP-hard and propose a three-stage heuristic for calculating the sojourn time at each location for K mobile sinks. In the first stage, the sojourn time profile at each h-feasible configuration is determined using a linear program. Then, optimal trajectories for each mobile sink are built via greedily adding h-feasible configuration if the *benefit* to the network lifetime is maximized. Finally, the exact sojourn time in each configuration is calculated using another linear program. The experimental results show that the proposed algorithm performs at up-to 93% of the optimal solution.

In [70], the authors propose a multi-sink relocation algorithm with several components. They adopt a centralized approach for the routing methodology. Link weights are computed using a cost function, and finally a mathematical model is given for determining optimal sink positions, i.e., network configuration. The authors use a local search algorithm for determining local optima in the solution space. The experimental results are presented around emphasizing mobility utilization compared to stationary sinks.

Chapter 3

Traffic and Energy-Load–Based Sink Mobility Algorithms for Wireless Sensor Networks

Sink mobility is among the most effective solutions to improve network lifetime [71]. This solution aims to solve the *hot-spot* problem, in which a sink's first-hop neighborhood suffers from quick energy depletion due to high rate of message forwarding [16]. In order to distribute the load among sensor nodes in the network, it is common in the literature to move sinks (base stations) and change their first-hop neighbors periodically to balance energy consumption as much as possible.

In this chapter, we propose time and space efficient algorithms that incorporate different parameters into account while determining the next position the sink has to move. More specifically, our proposed algorithms consider parameters like distance among nodes or packet loads of nodes, and collect these parameter values into a matrix structure, i.e., a table, to schedule the movement of the sink.

Without loss of generality, we use tree-based deterministic routing such that each node sends its own packet and relays its children's packets to the same parent

for a given sink site. Necessary parameters are learned by the sink node with a *training* phase so that the same routing tree is constructed in each visit of the sink node to the same location. This method enables us to construct a packet traffic load matrix structure T of sink locations (rows) and nodes (columns) such that each entry (t_{ij}) in the matrix lists how many packets node j would send and relay for a given sink location i. This brings us to our first approach, the packet-load based movement algorithm (PLMA). This algorithm basically uses and updates the packet traffic load matrix to choose the next sink site.

In our second approach, i.e., in energy-load–based movement algorithm (ELMA), we use computed energy expenditure values in the matrix instead of only packet load values. This model is more accurate than the PLMA, since it differentiates between sending the same number of packets to a farther or nearer parent. Otherwise it uses the same approach as the PLMA in choosing the next sink site. We also give an IP formulation for the problem and we use its solution as a benchmark to judge the performance of our algorithms.

We performed extensive simulation experiments to compare our algorithms with random sink movement (RAND), with a min-max (MM) approach, in which the sink learns about the minimum remaining energy values of each site and moves to the one with the maximum energy, and static sink case (STS). The results show that ELMA algorithm performs up to 80% better than the random movement and 100% better than min-max approaches. It also provides more than a five times longer lifetime compared to the static sink case. ELMA is just below five percent of the optimal value, however, it has better latency compared to the optimal (OPT) and has shorter running time. Lastly, we adapt the solution to the multiple-sink case, where we have multiple mobile sinks that can move to utilize energy in a better manner. In this case, the rows of the matrix become possible combinations of sink sites. Our simulation results for this case show that ELMA algorithm performs better than the random movement method by up to a factor of 2.1. This ratio increases to 2.6 when the nodes are not uniformly deployed.

The rest of the chapter is organized as follows: In Section 3.1, we give the system model. We present our algorithms in Section 3.2 and 3.3. In Section 3.4,

we give and discuss our performance evaluation results. Finally, we conclude the chapter in Section 3.5.

3.1 System Model

We consider a wireless sensor network that has n static sensor nodes and a mobile base station (sink). We assume there are p different site alternatives that the sink node can visit and stay. Sensor nodes are deployed to the region of interest in a random manner. There is a *training* phase, as in [68], such that mobile sink visits all sink sites once before network starts its operation. After the mobile sink moves to a location i, it stays there for a while and constructs a routing tree by initiating the flooding of a control packet in the network. Each node receiving the packet sets its parent and re-broadcasts the packet after adding its ID to the packet. In this way a routing tree is formed. Each node saves this parent ID for that specific sink location to transmit its packets every time sink visits that location. Nodes also notify their parents about their decisions so that parents can calculate how many packets they will relay when sink is at that location i. Each node sends this packet count information to the sink during the training phase. Sink uses these values to construct matrix T, either directly for packet load matrix, or applying the energy model to calculate energy expenditure to obtain energy load matrix. When sink moves to a new location it does not initialize routing tree reconstruction process, instead sends the sink location ID to the nodes. Since nodes know which parent to select for each sink location, they send the packets destined to this node while sink stays at that location.

After tree formation, nodes start to sense the environment. Each sensor node generates packets with a rate Q. We assume that each sensor node has enough buffer size to avoid losing packets during the travel of the sink from the current site to the next one, or this time is ignored as in [21,22,40]. In this work, we define network lifetime as the period of time until the first node dies, which is a common definition in the literature.

3.2 Proposed Algorithms

As mentioned in the previous section, sink node moves to each possible sink migration point and constructs a tree-based routing topology rooted at itself for each point during *training* phase. In this way, a packet-load matrix T is constructed such that t_{ij} is the number of packets that node j (o_j) would send when the sink is at migration point i (p_i).

$$T = \begin{array}{c} \\ p_1 \\ p_2 \\ \vdots \\ p_p \end{array} \begin{array}{c} \begin{array}{cccc} o_1 & o_2 & \dots & o_n \end{array} \\ \begin{pmatrix} t_{11} & t_{12} & \dots & t_{1n} \\ t_{21} & t_{22} & \dots & t_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ t_{p1} & t_{p2} & \dots & t_{pn} \end{pmatrix} \end{array}$$

Before this matrix is constructed, we need to know the possible sink migration positions, which can be determined by various algorithms [72]. For example, the region can be considered a grid and each grid cell can be a migration point. Alternatively, neighborhood information can be used in determining possible migration points, moving the sink to dense neighborhoods, for example. Once the migration points are determined, there will be one row in the matrix for each possible migration point. For each migration point, we compute the possible routing tree rooted at that point and compute what the packet-load of each node will be in the tree.

After these computations we know for each point i and node j what the packet load of the node, i.e., t_{ij} value of the matrix, will be at that point. In this way we can construct the load matrix and the sink can schedule its movement before network operation begins. Pre-scheduling the movement is useful in reducing the time needed to travel to the next sink position, hence reducing the buffer requirement at sensor nodes and reducing the delay that packets will experience.

We also modify the algorithm to consider not only the packet load on a node but also the energy cost of transmitting those packets to the next node, i.e., the

parent. Hence, our second algorithm is based on nodes' energy expenditures for a given sink position.

3.2.1 Packet-Load Based Movement Algorithm (PLMA)

Our first algorithm aims to minimize the maximum number of packets sent by a node in each round. A round is the time duration the sink stays constant at a position (migration point). The pseudo-code of our algorithm is given in Algorithm 1.

In the algorithm, after we determine the possible migration points, the sink constructs a routing tree for each sink candidate position and calculates the number of packets that each node will send to its parent in each round. In this way, the packet-load matrix T is obtained. After that, the algorithm determines the maximum value in each row, selects the row with the minimum value, and takes this row as the *current packet list (cpl)*. We call this min-max search. The selected row corresponds to the position to move next. After the sink migrates to that point, it will operate there for fixed number of rounds or until a certain amount of change in the energy of its first-hop neighbors is detected. Then the sink updates matrix T by adding the current packet list value to each row of T and in this way obtains a new matrix U each time it makes a decision to move. Values in each row i of matrix U estimate the packet loads of the nodes if sink moves to location i. Sink again runs the min-max search over U to determine the new packet list value. This iteration continues until a node depletes its energy. Sink keeps both matrix T and U to determine next location to move. T is constructed before network starts its operation and not changed. U is calculated each time sink makes a decision by adding *cpl* to each row of T. The selected row of U becomes the new *cpl* and it will be used to calculate matrix U when the next decision time comes.

The algorithm has $O(np)$ preprocessing time complexity to insert the data into the matrix. In each round, we need $O(p)$ operations to determine the sink's next migration point.

Algorithm 1 Packet-Load–Based Movement Algorithm (PLMA)

Input: Node coordinates and transmission range
Output: Schedule of sink movements
procedure PLMA(c, t_x) ▷ c: node coordinates, t_x: transmission range
    $p \leftarrow getMigrationPoints(c, t_x)$
    for $i \leftarrow 1, size(p)$ **do**
        $tt \leftarrow constructTopology(c, t_x, p_i)$
        for $j \leftarrow 1, n$ **do** ▷ for each node
            $t_{ij} \leftarrow k_i$ ▷ k_i: # of packets to send for o_j
        end for
    end for
    for $i \leftarrow 1, size(p)$ **do**
        $mp(i) \leftarrow max(t_i)$ ▷ max element for each row
    end for
    $r \leftarrow min(mp)$ ▷ index of minimum of maximums
    $cpl \leftarrow t_r$ ▷ initialize current packet list
    while $e > 0$ **do** ▷ all nodes are alive
        for each round **do**
            for $i \leftarrow 1, size(p)$ **do**
                $u_i \leftarrow t_i + cpl$ ▷ current packet matrix
            end for
            for $i \leftarrow 1, size(p)$ **do**
                $mp(i) \leftarrow max(u_i)$ ▷ max for each row
            end for
            $r \leftarrow min(mp)$ ▷ index of min of max
            $cpl \leftarrow u_r$ ▷ update current packet list
        end for
    end while
end procedure

Each node keeps parent IDs for each sink location. Since node IDs can be represented in $log(n)$ bits and there are p locations in the area, space complexity becomes $p\,log(n)$ bits. For 500 nodes and 30 sink sites, it requires around 34 bytes to keep this array. This is acceptable since most of the nodes have more than 32 KB program and data memory [73]. Sink maintains a matrix of $p \times n$, which requires $O(p\,n\,log(n))$ space. For 500 nodes and 30 sink sites, it requires around 16 KB memory to store this matrix. Sink nodes have more resources than sensor nodes. For instance, a sink acting as a gateway can have more than 1 GB RAM [74].

3.2.2 Energy-Load–Based Movement Algorithm (ELMA)

Our second algorithm is a slight modification of the first one in that it uses the computed energy load/expenditure of each node instead of the packet load. The second algorithm constructs again a matrix T, but this time, for each candidate sink point i, i.e., for each row, it computes how much energy the nodes would consume to transmit their packets to the next node in the routing tree for that sink position i. Each node piggybacks the remaining energy value to the sink node periodically. Sink stores these values in a residual energy row e. It subtracts each row i of matrix T from e to construct row i of matrix U. So, each row i of matrix U estimates nodes' remaining energy values if the sink would go to site i in the next round. Hence, our second algorithm is energy based, where energy consumption is considered to be related both to transmitted packet count and to where the packets are transmitted.

ELMA (see Algorithm 2) becomes the *dual* of our first algorithm such that we should use max instead of min (and vice versa) and subtraction instead of addition. Using the energy-based approach is more meaningful if it is possible for a sensor node not to send the expected number of packets in each round. For example, some nodes may send more packets if an important event is detected, or some packets might be lost due to some node or link failures in the network. This model is more accurate than the packet-based approach because it takes

distance into consideration as part of the energy consumption model when sending a packet. When a node sends, say two packets, to different parents for different sink sites, the energy-based model has different values for each sink site, while the packet-based scheme has the same value (namely two).

Algorithm 2 Energy-Load Based Movement Algorithm (ELMA)

```
1: Input: Node coordinates and transmission range
2: Output: Schedule of sink movements
3: procedure ELMA(c, t_x)        ▷ c: node coordinates, t_x: transmission range
4:     p ← getMigrationPoints(c, t_x)
5:     for i ← 1, size(p) do
6:         tt ← constructTopology(c, t_x, p_i)
7:         for j ← 1, n do                                   ▷ for each node
8:             t_ij ← tx_i + rc_i                            ▷ t_x and r_x cost for o_j
9:         end for
10:    end for
11:    for i ← 1, size(p) do
12:        mp(i) ← max(t_i)                                  ▷ max element for each row
13:    end for
14:    r ← min(mp)                                           ▷ index of minimum of maximums
15:    s_c ← p_r                                             ▷ set sink's next coordinate
16:    while e > 0 do                                        ▷ all nodes are alive
17:        for each round do
18:            u ← e − t                                     ▷ update current energy matrix
19:            for i ← 1, size(p) do
20:                mp(i) ← min(u_i)                          ▷ min for each row
21:            end for
22:            r ← max(mp)                                   ▷ index of max of mins
23:            s_c ← u_r                                     ▷ set sink's next coordinate
24:        end for
25:    end while
26: end procedure
```

Like our first algorithm, our second algorithm requires $O(np)$ preprocessing time to initialize the matrix with the energy loads of nodes at possible migration points. It needs $O(p)$ time in each round to determine the next migration point for the sink. Sensor nodes' overhead is same as in PLMA case. However, sink has to consume more space to keep the energy matrix. It requires $p\,N\,log(E)$ bits, where E is the initial energy value in microjoule. For 500 nodes, 30 sink sites, and 10K microjoule it requires around 26 KB to store energy matrix.

3.2.3 Mathematical Model

We can consider this problem as a variant of the $0-1$ *knapsack* problem. In this case, we want to visit each sink site (add item), each of which has a different value (energy consumption) in each dimension (node), as much as possible without exceeding the initial available energy value (knapsack capacity). Each sink site corresponds to a knapsack item, and each node represents a dimension of that item. In this way, the problem becomes an instance of the *unbounded multi-dimensional knapsack* problem [75]. In this version, each dimension (node) has the same limit (energy), and all profit values are the same (staying the equal number of rounds x_i at different sink sites contributes equally to the network lifetime).

We can formulate the problem also as an IP formulation:

$$\begin{aligned} \max \quad & \sum_{i=1}^{m} x_i \\ \text{s.t.} \quad & \sum_{i=1}^{m} t_{ij} x_i \leq e, \quad j = 1, \ldots, n \\ & x_i \geq 0, \quad x_i \text{ integer, for all } \quad i = 1, \ldots, m \end{aligned} \tag{3.1}$$

It is shown that solving the unbounded multi-dimensional knapsack problem is *NP-complete* [76]. Finding a fully polynomial-time approximation scheme (FPTAS) for the problem exhibits the same hardness. We solved the above-formulated problem using the MATLAB CPLEX solver for obtaining optimal values for small networks to see how close our heuristics algorithms are to the optimum.

3.3 Multiple-Sink Case

We can also adapt our algorithm to sensor networks with multiple sinks. Instead of representing a single sink site, one row of the load matrix can be a combination of sink placements. For each combination of possible sink placement, sinks move to these points and routing trees are established using the approach mentioned in Section 3.1, with the only difference that each sink initiates a parallel broadcast process independent from other sinks. Each sensor node chooses the nearest sink and the shortest path to that sink after receiving the broadcast packets. At the end of this process, we have s different mutually exclusive (each node is connected to only one sink) and collectively exhaustive (no sensor node is disconnected) trees rooted at sink nodes.

If we have p migration points and s sinks in the environment, then the rows in matrix T become the enumeration of all possible $\binom{p}{s}$ sink placements:

$$T = \begin{array}{l} \\ \{p_1, \ldots p_{s-1}, p_s\} \\ \{p_1, \ldots p_{s-1}, p_{s+1}\} \\ \vdots \\ \{p_{m-s+1}, \ldots p_{m-1}, p_m\} \end{array} \begin{array}{c} \begin{array}{cccc} o_1 & o_2 & \ldots & o_n \end{array} \\ \begin{pmatrix} t_{11} & t_{12} & \ldots & t_{1n} \\ t_{21} & t_{22} & \ldots & t_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ t_{m1} & t_{m2} & \ldots & t_{mn} \end{pmatrix} \end{array}$$

The number of rows in matrix T is no longer m, but the number of possible combinations of $\binom{p}{s} = \frac{p!}{s!(p-s)!}$. The values in the matrix can represent the packet load or energy load on the nodes for a given position of sinks. We can apply our previous algorithms (PLMA and ELMA) to the new matrix for solutions to the multi-sink case.

The number of rows in the load matrix for the multiple-sinks case can be approximated by Stirling's formula: $\sqrt{2\pi p}(\frac{p}{e})^p$. Therefore, our multi-sink algorithms have $O(np^p)$ pre-processing time complexity to initialize the matrix. In each round (before each move), we need $O(p^p)$ operations to determine the next sink positions.

Obviously, using these algorithms for large values of p and n is not feasible. However, they can be used for cases where the $\binom{p}{s}$ value is less than a threshold. We call this as *binomial threshold* (t_b). What t_b will be is a design decision and should be determined by considering the storage capacity of sensor nodes to store the matrix. For a high number of sink sites, we can choose a subset of migration points. The size of that subset, th_s, should be the maximum integer where $\binom{th_s}{s} \leq t_b$), The subset can be selected by using, for example, the *k-means* algorithm. Then we can apply our PLMA or ELMA algorithm to this reduced matrix. We call these multi-sink algorithms that work with fewer combinations as MS-PLMA (Algorithm 3) and MS-ELMA (Algorithm 4), and show their pseudo-codes below.

Algorithm 3 Multiple Sinks Packet-Load–Based Movement Algorithm (MS-PLMA)

1: **procedure** MS-PLMA(c, t_x) ▷ c: node coordinates, t_x: transmission range
2: $p \leftarrow getMigrationPoints(c, t_x)$
3: **if** $\binom{p}{s} > t_b$ **then** ▷ t_b: binomial threshold
4: $p' = kmeans(p, th_s)$ ▷ th_s max int, $\binom{th_s}{s} \leq t_b$
5: **end if**
6: $PLMA(c, t_x, \binom{p'}{s})$
7: **end procedure**

Algorithm 4 Multiple Sinks Energy-Load–Based Movement Algorithm (MS-ELMA)

1: **procedure** MS-ELMA(c, t_x) ▷ c: node coordinates, t_x: transmission range
2: $p \leftarrow getMigrationPoints(c, t_x)$
3: **if** $\binom{p}{s} > t_b$ **then** ▷ t_b: binomial threshold
4: $p' = kmeans(p, th_s)$ ▷ th_s max int, $\binom{th_s}{s} \leq t_b$
5: **end if**
6: $ELMA(c, t_x, \binom{p'}{s})$
7: **end procedure**

3.4 Performance Evaluation

We implemented and simulated our algorithms in MATLAB to evaluate and compare them with some other approaches. In this section we discuss the results

of these simulation experiments. We compared our proposed algorithms (ELMA, PLMA, and their multiple sinks variants) against the optimal case (OPT) and other approaches (MM, RAND, and STS). Below we briefly describe all simulated and compared methods.

- OPT: Sink moves to the migration points and stays there according to the results given by the optimal model.
- ELMA: Sink visits sites as specified by our Algorithm 2.
- PLMA: Sink moves according to the results given by our Algorithm 1.
- MM: Minimum energy values of sink's first-hop neighbors are collected from each site and sink moves to the one with maximum energy among them.
- RAND: Sink selects a sink site randomly and moves to it.
- STS: Sink does not move but stays static in the center of the area.

We used the following two metrics in comparing these methods:

- *Network lifetime*: time until the first sensor node depletes its energy. This is a commonly used network lifetime definition.
- *Latency*: average hop count that a packet travels until it reaches the mobile sink node. We model latency as the number of hops traveled.
- *Total distance*: Total distance that mobile sink travels during the network lifetime.

3.4.1 Simulation Parameters

The sensor networks generated in our simulations have n static sensor nodes and one mobile sink. The sensor nodes are deployed randomly to a region of interest

(if not stated otherwise). Square-shaped regions are used in the simulations, which are generally of size either 300×300 m^2 or 400×400 m^2. After the mobile sink moves to its initial location, it broadcasts its location and nodes select the previously saved parent id (learnt from the *training* phase) to send their messages (topology is constructed). After the topology construction, nodes start sensing the environment. There is a constant packet generation rate Q (1 packet/s) for each sensor node $i \in N$. Binomial threshold is set to 4100 for multi-sink experiments.

The energy model and the radio characteristics used in the simulations originate from [77]. Transmission energy cost depends on packet size (number of bits in a packet) and the square of the distance between the transmitting and receiving nodes. The received energy cost is related only to the packet size. We assume data packets are 50 bytes long and control packets (tree establishment packets) are 20 bytes long. We assume the radio dissipates $E_{elec} = 50$ nJ/bit to run the transceiver circuitry and $\epsilon_{amp} = 100$ pJ/bit/m^2 for the transmit amplifier to achieve an acceptable $\frac{E_b}{E_n}$ [77]. Each sensor node has energy of 10 J initially. Energy information can be represented with fourteen bits and can be carried to the sink as piggybacked information in data packets. We use neighborhood-based sink-site determination algorithm to determine possible migration points that sink can move [72]. Migration points are selected with this algorithm such that neighbors of these selected points will cover the all the nodes in the area. Migration points, shown as asterisks, selected by this algorithm for a sample topology with 400 nodes are given in Figure 3.1. The parameters of our simulations and their typical values are summarized in Table 5.1.

All our simulations are done in the MATLAB environment. Our IP model is solved with CPLEX Optimizer supplied through IBM Academic Initiative program [78]. The CPLEX for the MATLAB feature of this software provides an API, which helps solve the problem in the same MATLAB simulation environment.

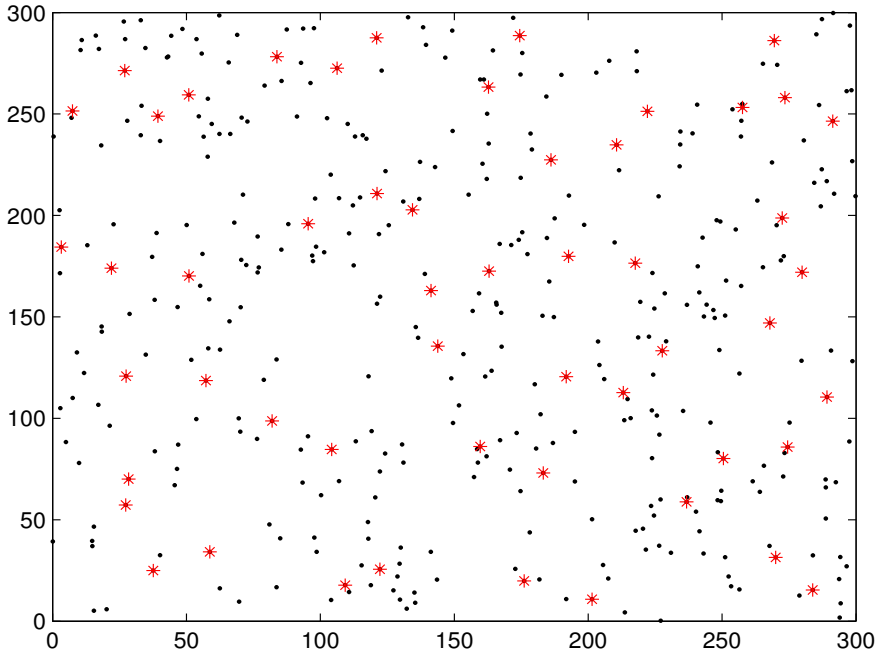

Figure 3.1: Neighbor based sink-site determination.

Parameter	**Value**
Area	$300 \times 300\ m^2$
Number of Sensor Nodes	400, 500, 600, 700, 800
Node Deployment	random and uniform
Transmission Range	25 m
Data Routing	Tree based [72]
Sink Site Determination	Neighborhood SSDA [72]
Nodes' Initial Energy	10 J
Radio Characteristics	First-order radio model [77]

Table 3.1: Simulation parameters

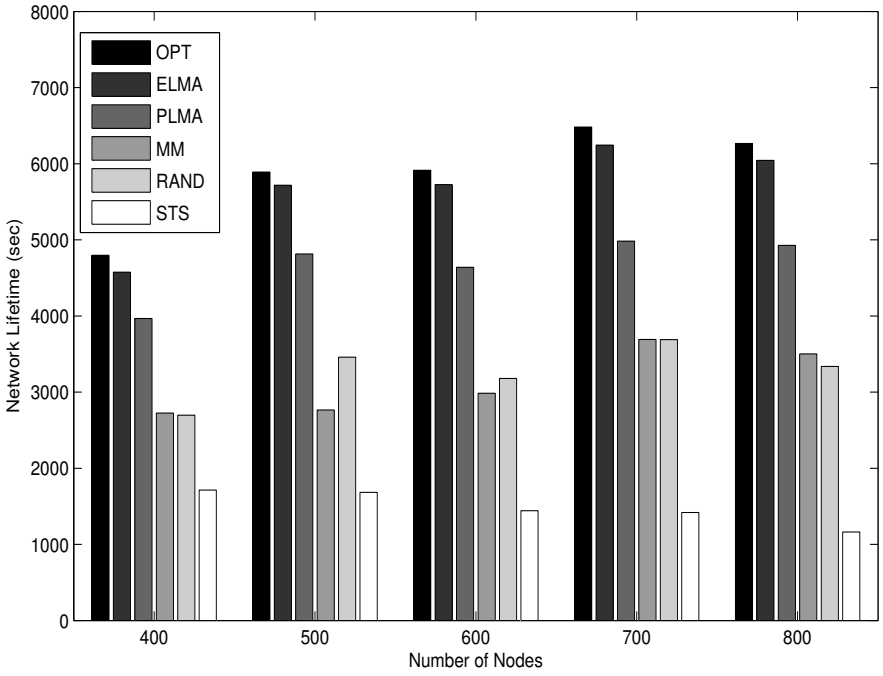

Figure 3.2: Network lifetime versus number of nodes.

3.4.2 Simulation Results

Network lifetime values of the various schemes for number of nodes between 400 and 800 are given in Figure 3.2. Transmission range is fixed to 25 m. As the figure shows, the optimal lifetime is just three to five percent more than the lifetime achieved with our energy-based algorithm (ELMA). Our ELMA improves the random movement scheme by 65 to 81%, and performs up to 5.2 times better than the static sink case, and it performs around 20% better than our packet-based approach (PLMA).

We also tested the algorithms' performance for the skewed deployment of nodes to a region, where nodes are not uniformly distributed. Figure 3.3 shows the experiment results for various number of nodes, which although have similarity to previous results, exhibit two main differences: 1) Random movement performs worse in skew deployment compared to uniform deployment, because the random points may have been poorly selected and may reside in a sparse area, for example.

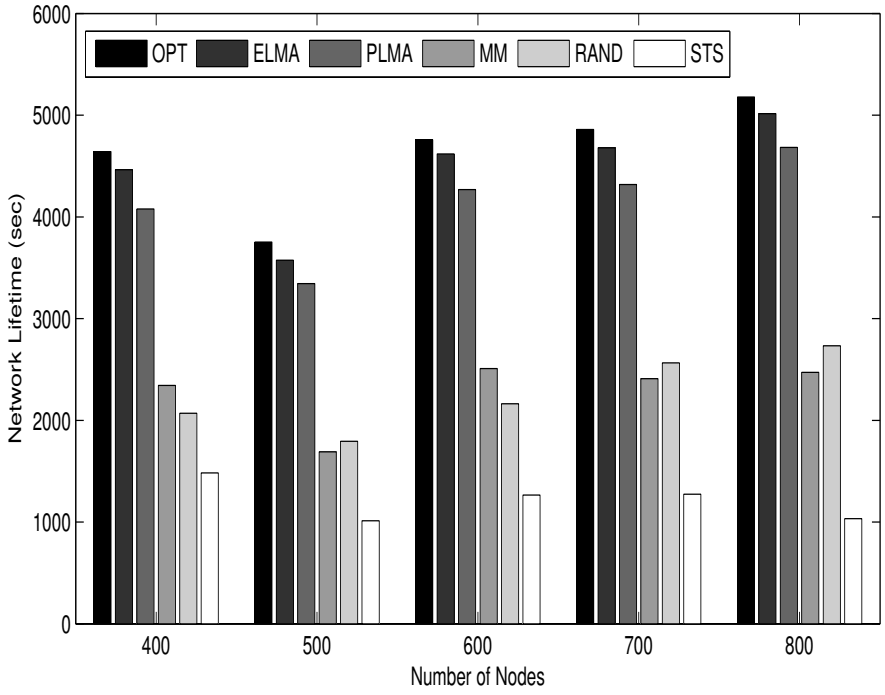

Figure 3.3: Network lifetime versus number of nodes under skew deployment.

Our ELMA algorithm performs at most 80% better in uniform deployment, where it performs more than 100% better in skew deployment. 2) The performance is not directly proportional to the number of nodes when the nodes are deployed in a skewed manner. In general, the lifetime is shorter with skew deployment compared to uniform deployment.

In Figure 3.4, we investigate network lifetime versus transmission range. The transmission range varies between 25 m and 40 m and the number of nodes is fixed to 400. We see results similar to previous figure; however, the performance difference between the static and mobile sink cases is not as great as when the number of nodes is varied. When transmission range increases, packets reach the sink node with fewer hops, which reduces the load on the one-hop neighbors, making sink movement less effective.

Latency (average hop count) values of the schemes for different transmission range values are presented in Figure 3.5. Latency values of the ELMA and PLMA

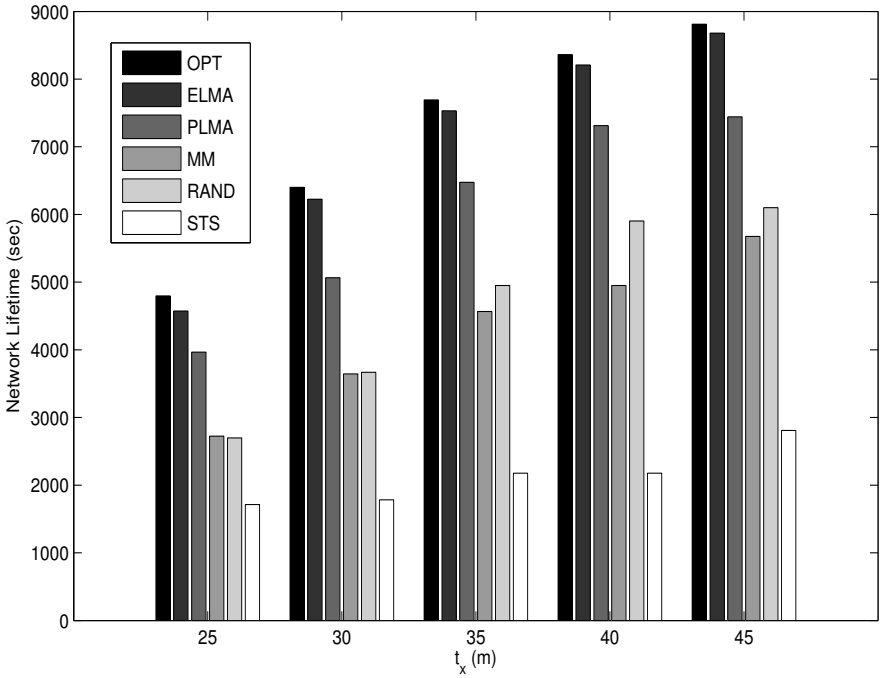

Figure 3.4: Network lifetime versus various transmission ranges.

and the OPT case are very close to each other, while the random scheme has slightly better latency (around five percent lower). The static sink case has the best latency values (around 33% lower), because the sink is always placed in the center of the area, which minimizes the average hop count.

Average running time values for the OPT and ELMA are given in Table 3.2. The IP (OPT) running time increases dramatically when the number of nodes, hence the number of constraints, increases. For example, it takes around 136 times longer to execute when the number of nodes is 800. Solving the IP takes sometimes more than 7 hours. There are also many runs in which CPLEX does not terminate normally, but exits with an *out of memory* error. One could solve the IP problem on a more powerful machine and then upload the result to the sink node before the network starts operating, but the results would take a very long time to get, with a benefit of less than five percent lifetime improvement compared to ELMA algorithm.

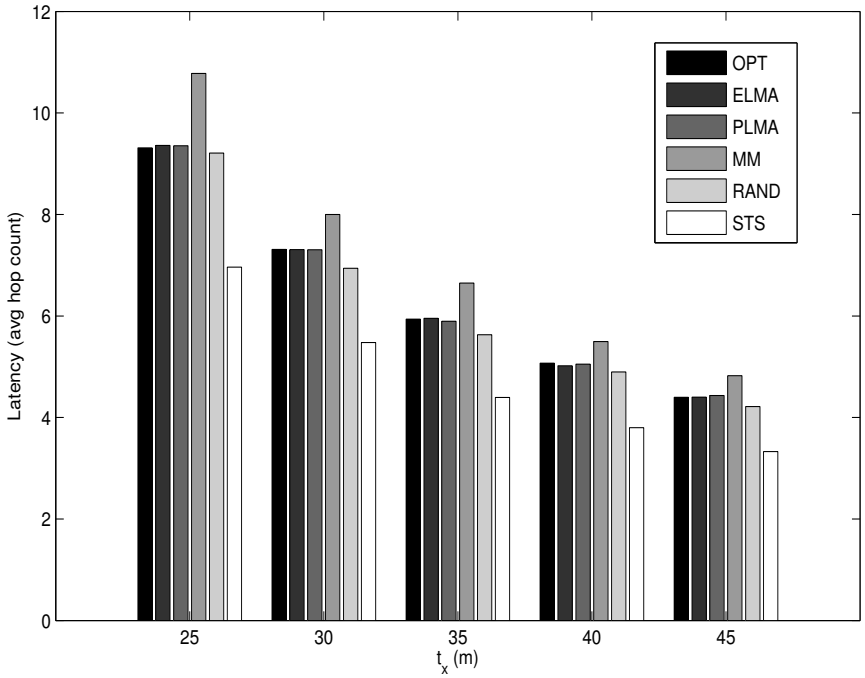

Figure 3.5: Latency versus transmission ranges.

The optimal scheme uses the *energy* table, calculated prior to network operation, which in theory means that every node's number of packets to transmit is known and does not change during operation. However, this is not always the case; sometimes nodes send extra packets (retransmissions) when there are collisions in the environment. The optimal algorithm cannot adapt to such a case because it is not dynamic, using only prior information. The ELMA uses nodes' residual energies (which are piggybacked in the data packets), so if there is a deviation regarding about energy expenditure expectations (from packet retransmissions, for instance), then the algorithm should adapt it in the next round. ELMA algorithm runs faster, can adapt to cases when unexpected packet transmissions occur, and sacrifices less than five percent of the network lifetime in doing so.

Network lifetime values for higher numbers of nodes (without optimal values) are given in Figure 3.6. Sink mobility efficiency increases directly proportional to the number of nodes. The energy-based approach has 3.2 times more network

	Number of Nodes				
	400	500	600	700	800
OPT	33	383	536	619	5575
ELMA	14	20	26	34	41

Table 3.2: Average running times (in seconds) of the two approaches.

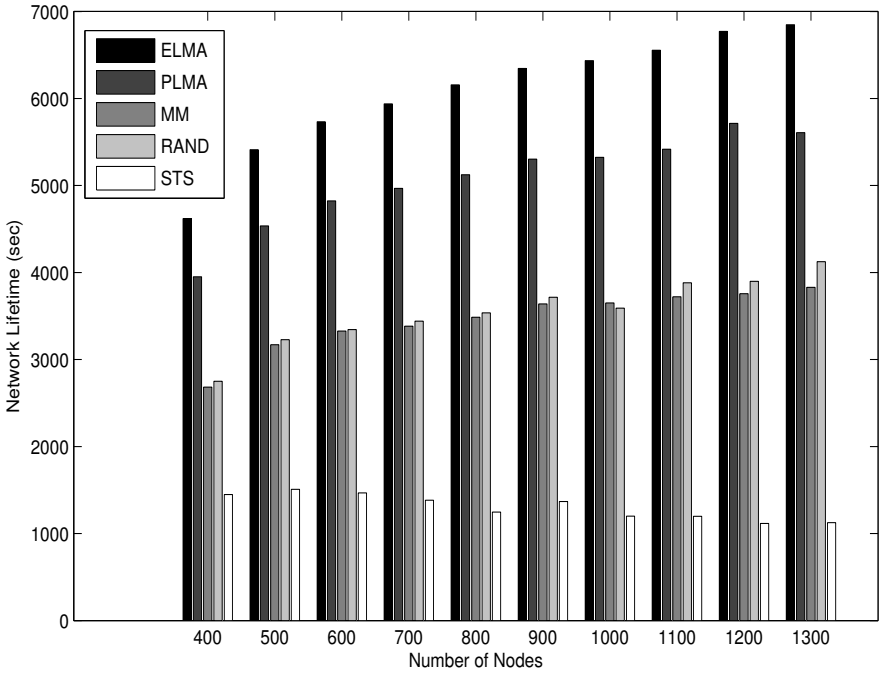

Figure 3.6: Network lifetime versus number of nodes.

lifetime when the number of nodes is 400. However, this ratio increases to six when there are 1300 nodes in the area.

Network lifetime values for different numbers of sinks are given in Figure 3.7. The number of sink sites is fixed to 15. Since the number of all possible sink placements is in the order of thousands, we cannot run the optimal algorithm due to its long running times. For the static case, we run the *k-means* clustering algorithm and places for sinks are determined using the output of clustering. Placing multiple sinks optimally in the area is another research issue and beyond the scope of our work. Results show that our MS-ELMA performs better than random movement by a factor of up to 2.15, and better than the static sink case by

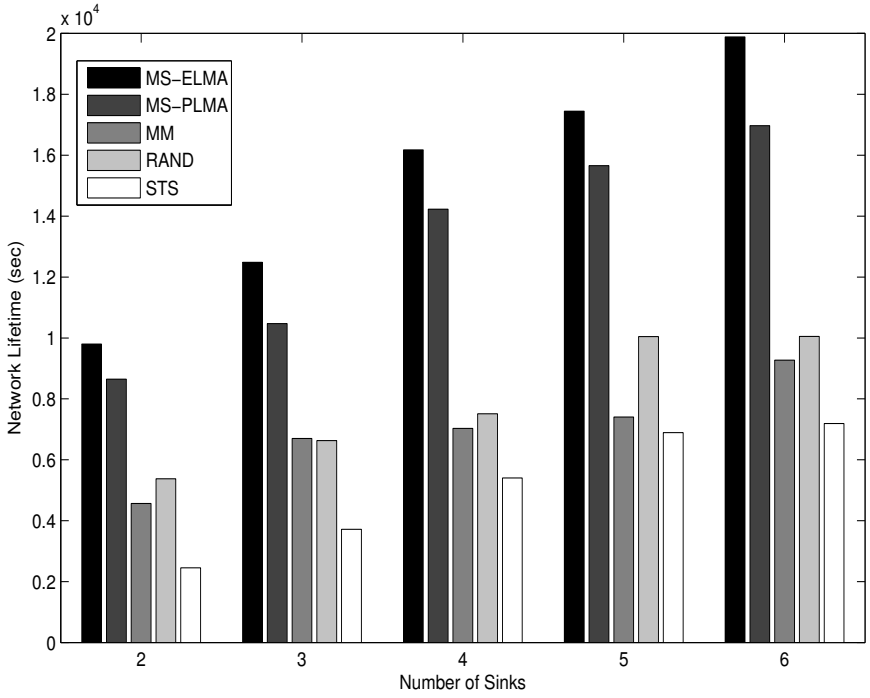

Figure 3.7: Network lifetime versus number of sinks.

a factor of up to four. We have slightly worse improvement ratio against the static sink approach because there are multiple sinks and thus the traffic load of sensor nodes is more balanced than in the single-sink case. This reduces the benefit of mobility because the main goal is to decrease unbalanced load distribution among sensor nodes.

Latency values for different number of sinks are given in Figure 3.8. The MS-ELMA and MS-PLMA have almost the same latency (they differ by at most three percent). These algorithms have lower values compared to the min-max (15% on the average) and random movement (six percent on the average) algorithms. Static sinks have the lowest latency values, as in the single-sink case, however static sink performs 17% lower on average, while in the single sink case it performs about 28% better. Because, sinks are not placed optimally in the static case, it has a poorer latency performance.

Network lifetime values for different numbers of sink sites are presented in

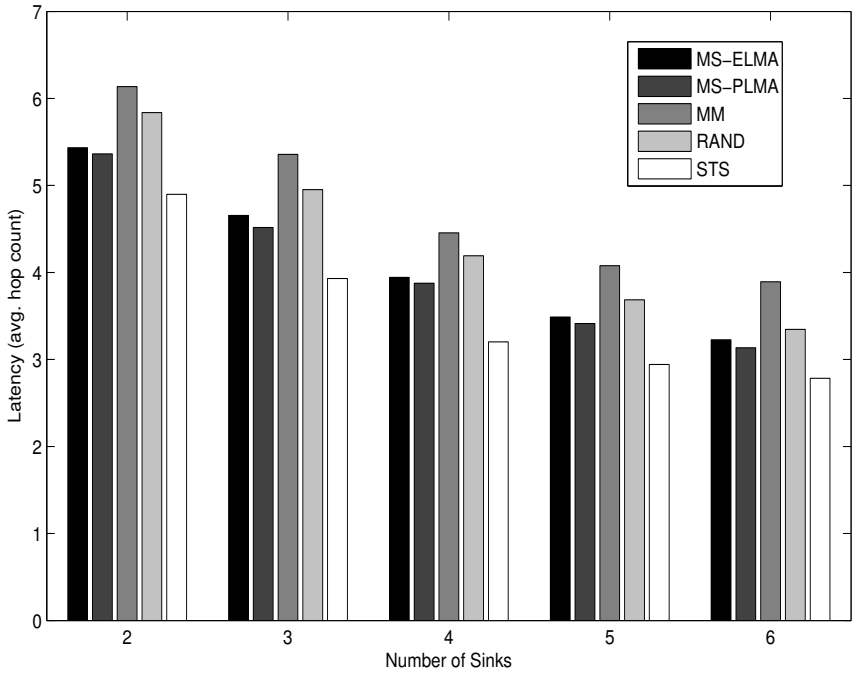

Figure 3.8: Latency versus number of sinks.

Figure 3.9. The number of sinks is fixed to three in the experiment. Networks have better lifetime values when the number of sink sites increases because there are more options to consider when moving the sinks. The relative performance of the MS-ELMA to the random movement algorithm is also affected by the number of sink sites. The MS-ELMA performs better than the random movement by a factor of 1.5 when there are 15 sites (which also means 455 different combinations in which to place three sinks), and by a factor of 2.15 when there are 30 sites (4060 different combinations).

Network lifetime values for different numbers of sinks under skew deployment are given in Figure 3.10. The MS-ELMA gives relatively better performance against the random movement compared to the uniform distribution. It performs better than random movement by a factor of 2.6, and by 2.15 when nodes are randomly and uniformly deployed.

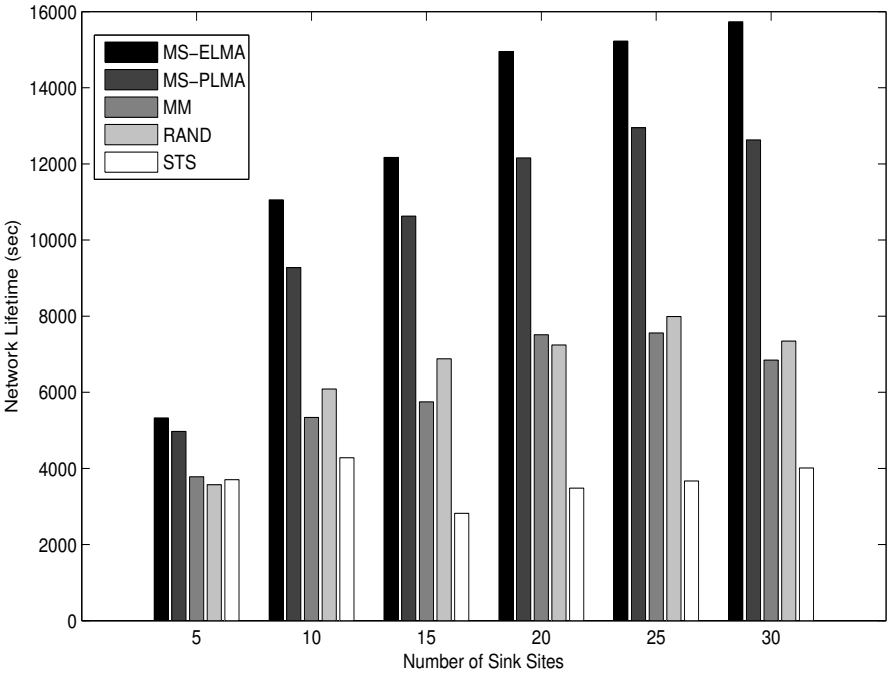

Figure 3.9: Network lifetime versus number of sink sites.

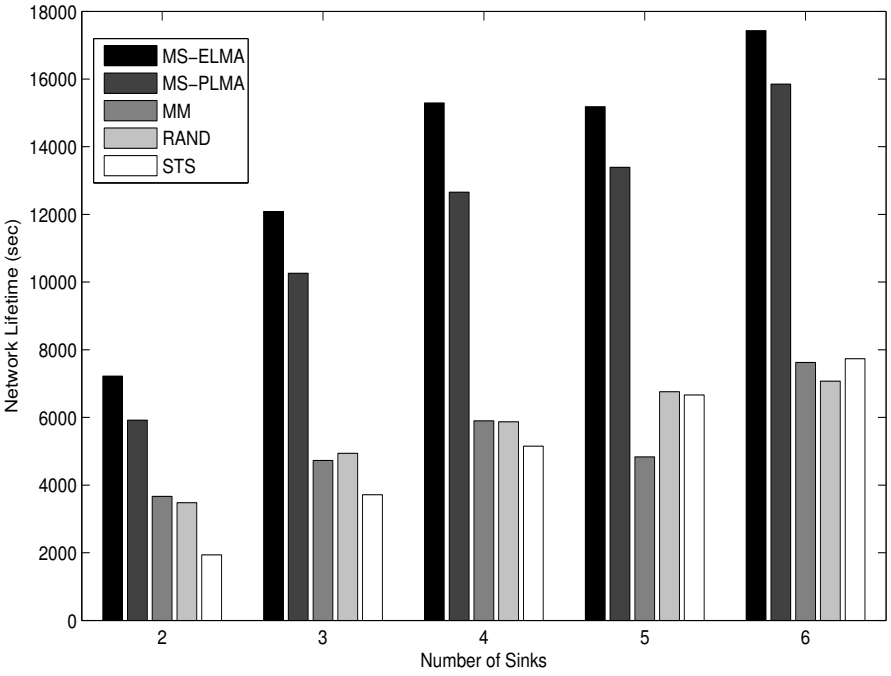

Figure 3.10: Network lifetime versus number of sinks under skew deployment.

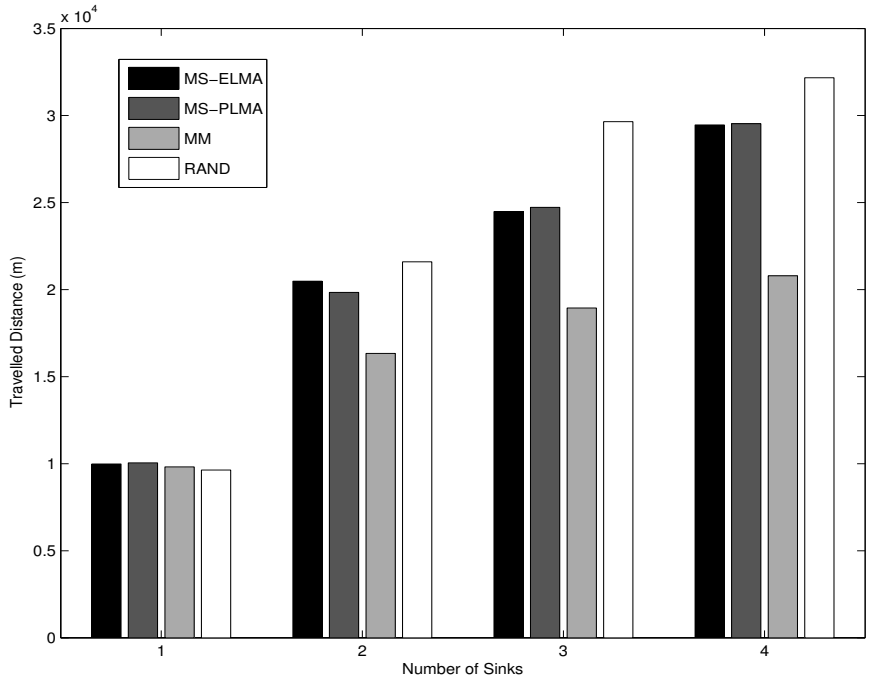

Figure 3.11: Distance traveled versus number of sinks.

Distance-traveled values for different number of sinks under random deployment are given in Figure 3.11. Sinks traveled a minimum amount of distance using the min-max algorithm (24% less compared to others on the average). The MS-ELMA and MS-PLMA have similar values and result in less distance traveled (seven percent less) compared to the random movement algorithm.

3.5 Conclusion

In this chapter, we propose mobile-sink algorithms, which consider nodes' packets or energy loads in deciding on the next place to move the sink. The objective is to distribute the load on sensor nodes in a balanced manner so that network lifetime is improved as much as possible. Given a routing tree rooted at the sink node, the number of packets each node has to send can be calculated and a load matrix can be obtained, which shows how much load a node has for each possible sink

position. We propose a packet-load–based algorithm (considering the packets a node has to relay in a round) and an energy-load–based algorithm (considering the energy consumed by a node in a round). The algorithms greedily select the sink site that minimizes the maximum load on a sensor node. The problem can be formulated as an IP and the optimal movement strategy can be obtained by solving the IP model, but this would take too much time for large networks. Our algorithms, on the other hand, can be used for very large networks and can quickly provide a close-to-optimal solution.

Our simulation results show that our energy-load–based algorithm provides up to five times better network lifetime than a static sink and 2 times better network lifetime than random movement. It is only five percent below the optimal solution. Our method has almost the same average latency as the optimal one, but runs much faster. Our energy-load–based scheme can also adapt to changes in the expected number of packet transmissions in a node, which can happen due to packet collisions or packet corruptions.

We also extend our algorithms to use in multiple sinks case (MS-ELMA and MS-PLMA) by limiting the possible number of combinations. These algorithms also perform better than random movement (by a factor of 2.15 and 2.60 for random and uniform, and skewed deployment, respectively) and static sink cases (by a factor up to four). They also have lower latency (six percent on the average) and sinks using these algorithms travel less (seven percent) compared to random movement.

Chapter 4

Adaptive Routing Topology Construction and Sink Mobility Algorithms

In the previous chapter, we proposed two single-sink mobility algorithms: packet-load and energy-load based algorithms to coordinate sink movements. These algorithms rely on the matrices (either packet–load or energy–load matrices) constructed during routing tree formation phase. A sink node visits each location and constructs routing topologies (which are trees) in a training phase before network start its operation. Each node selects its parent and calculates the number of packets it has to send (or the amount of energy it needs to transmit those packets) for each different sink position. Packet- and energy-load matrices are constructed using these trees and used when deciding next points to move.

Although these algorithms improve network lifetime, it is assumed that the same tree is formed every time the sink visits the same location. However, it may not be always possible for a node to select the same node as the parent each time the sink moves to the same location. Besides that, when tree topologies are determined (to construct matrices) before network starts (like in ELMA and/or in PLMA algorithm), nodes' load balance is done according to the number of packets

they relay, since all nodes have equal (full) energy values, initially. Although energy map of the nodes continuously change, initially determined parent-child relationships and packet counts values do not change during network lifetime.

In this chapter, we propose an *adaptive* single-sink mobility algorithm that does not require any training phase and enables the routing trees to change during network lifetime. We also give two complementary routing-tree construction algorithms that can be used to build energy-efficient trees considering the remaining energy values of sensor nodes.

This chapter is organized as follows. We give energy-efficient routing topology construction algorithms in Section 4.1. Then we present our adaptive single-sink mobility algorithm. Our experimental results are given in Section 4.3. Finally, the chapter is concluded in Section 4.4.

4.1 Energy-Efficient Tree Topology Construction Algorithms

Each time sink sojourns at a location; it reconstructs a tree-based topology for routing the packets from nodes to the sink. Since packet transmission is the most energy consuming operation in sensor networks, constructing energy-efficient routing trees that consider current remaining energy values of nodes can help in balancing energy-load and improving network lifetime. In this section, we give our energy-efficient routing topology construction algorithms in detail, which provide a basis for our adaptive sink-mobility algorithms explained thereafter.

First, we present our centralized routing topology construction algorithm, called CTCA, which improves our previous Load Balanced Topology Construction Algorithm (LBTCA) [72] by updating packet loads using remaining energy values of sensor nodes in Algorithm 7. In CTCA, nodes' parents and packet-load values are determined by sink node after receiving related information from sensor nodes. We also give a fully distributed topology construction algorithm, called

DTCA, in which each node selects its parent individually and independently, without any message exchange between the neighbors and/or parents. In this way, the packet and energy loads of nodes are affected in a distributed manner.

Both of our algorithms start with a broadcast phase. In this phase, sink node initiates a topology construction process by flooding a packet to its first-hop neighbors. Each node receiving this message re-broadcasts the message so that all nodes in the network finally receive the message. During this process, each node sets its logical level (hop-count to sink) and determines a candidate parent-list. Since broadcast phase is common for both of our algorithms, first we give details about this phase and then present our algorithms.

Algorithm 5 Broadcast Phase

```
1: function BROADCAST(H, pl)
2:     for i ← 1, N do                                   ▷ variable initialization
3:         H(i) ← −1                                     ▷ logical level (hop distance to sink)
4:         pl(i) ← −1                                    ▷ candidate parent list
5:     end for
6:     H(0) ← 0                                          ▷ sink sets its H and pl values
7:     pl(0) ← NULL
8:     brcs(0, 0, e_max)                                 ▷ sink broadcasts its id and logical level
9:     while H < 0 do     ▷ broadcast until all nodes are reached or ttl expires
10:        if rcv_i(j) then                              ▷ if node i receives a msg from j
11:            if H_i < 0 then                           ▷ this is the first msg received
12:                H_i ← H_j + 1
13:                brcs(i, H_i, e_i)
14:            end if
15:            pl(i) ← pl(i) ∪ j                         ▷ updates parent list
16:        end if
17:    end while
18: end function
```

4.1.1 Broadcast Phase

When sink sojourns at a site, it first broadcasts a packet to construct a tree-topology which will be used until it decides to move again. This message includes the ID of the sender, the minimum hop-count of the sender to sink, and the

remaining energy value of the sender. The ID and hop-count values are set to zero, and the remaining energy value is set to a global maximum value (100% energy) when sink initially sends the message, i.e., when the sender is the sink node. When a sensor node receives such a message, it saves the sender ID and increments the hop-count by one. Then the node re-broadcasts the message that includes now its ID, the incremented hop-count value, and its remaining energy value.

Each node waits for a certain amount of time, called broadcast threshold (B), before re-broadcasting the tree-construction packet. During broadcast threshold time, a node can receive multiple data packets from its potential parents, neighbors, and potential children (nodes belong to different logical levels in the final topology tree). After this time expires, it selects the minimum hop-count among the received values, increments it by one and sets it as its final hop-count value. It also saves the node IDs that have minimum received hop count value as potential parent candidates and in this way obtains a candidate-list.

Algorithm 6 Load-Based Topology Construction Algorithm

```
1: function LBTCA(H, pl, pr, l_p)
2:     ml ← max(H)                                  ▷ get max level from leaves
3:     for lk ← ml, 1 do                                    ▷ start from leaves
4:         cnl ← find(H, k)                          ▷ find nodes have level of k
5:         spl ← sort(cnl, pl)          ▷ sort nodes according to # of parents
6:         for i ← 1, size(cnl) do
7:             if size(spl_i) = 1 then              ▷ assign unique option as parent
8:                 pr(i) ← spl_i(1)
9:             else
10:                for j ← 1, size(spl_i) do
11:                    r(j) ← e_j/cpl^2
12:                end for
13:                ind ← max(r)
14:                pr(i) ← spl_i(ind);
15:            end if
16:            l_p(pr(i)) += l_p(i)                 ▷ updates packet loads
17:        end for
18:    end for
19: end function
```

Forwarding of topology construction messages continues until hop-count

reaches time-to-leave value (H) or no packet is received in $2B$ time. The time-to-live value can be obtained by dividing the maximum distance between two points in the area to the transmission range. More specifically, if a and b are the side lengths of a rectangular region and t_x is the maximum transmission range, then the Equation 4.1 gives the time-to-leave value, i.e., the H parameter of the deployment.

$$H = \lceil \sqrt{(a^2 + b^2)}/t_x \rceil \tag{4.1}$$

At the end of the broadcast phase, nodes' logical levels (hop-counts to the sink node) and parent-lists are set. This information is used to determine the parent for each node using one of the algorithms we propose below. Details of the phase is given in Algorithm 5.

4.1.2 Centralized Topology Construction Algorithm (CTCA)

In centralized topology construction algorithms, the sink node determines the routing tree and notifies the sensor nodes about their parent IDs. For this, the sink node should have logical-level (hop-count) and candidate parent-list information for all sensor nodes. This information can be obtained in two different ways. In the first way, if sink knows the exact location (coordinates) of the nodes then it can calculate both hop-counts and neighbor relationships without a broadcast phase. However, since knowing the nodes' exact coordinates cannot be possible (or expensive operation) in most of the cases, we do not use this method in our algorithm. In the second way, also the case in our centralized algorithm, each node can set its logical-level and parent-list with a distributed broadcast phase (explained above) and send these values back to the sink using one of the candidate parents to initiate topology construction process.

Sink starts topology construction process after logical level and parent list information are collected from the sensor nodes. We extend our previous load-based

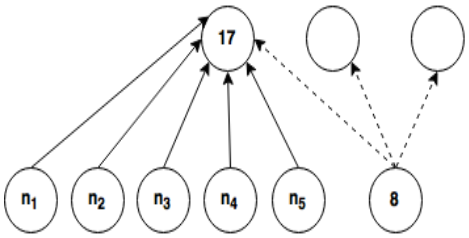

Figure 4.1: Parent selection in Load Based Topology Construction Algorithm.

topology construction algorithm (LBTCA) given in [72]. In LBTCA (Algorithm 6), tree construction process starts at the leaf nodes. Nodes are first sorted in a non-decreasing order according to number of candidate parents. The motivation behind this operation is to give priority to the nodes with few parent options to select their parents. Assigning these nodes' loads to the parents first causes the other nodes (with more parent options) to make decisions resulting efficient load balancing. For instance, assume that five different nodes $(n_1, n_2, \ldots, n_5)$ have only one (same) parent (say node 17) option to select (Figure 4.1). Another node (say node 8) has three different options including node 17 (which has the largest remaining energy), as well. If node 8 selects its parent before other five nodes, it will select node 17, and other five nodes will also select the same parent since they do not have any other option. However, if node 8 selects its parent after the nodes with less options, then it will probably select another parent because node 17 has already five-packet load.

Sink node selects parents considering the ratio of remaining energy to packet load of each sensor node, which can be expresses as e/cpl^2, where e is the remaining energy of a sensor node and cpl is the current packet load of the candidate parent. For each candidate parent for a sensor node, the sink calculates this ratio and selects the candidate with maximum value of this ratio as the parent of the node. This iteration continues for each level in the tree. When the operation is completed, packet-load in each level is balanced as much as possible.

Although LBTCA aims to balance the load, it considers just one level up when determining parents. However, there can be nodes with lower energy values in the upper levels that must forward more load than it can handle because of being the single parent candidate of a node with high energy. For instance, assume that

a node 10 has relatively high remaining energy value than the other candidates for five different nodes. These five nodes select node 10 as parent and node 10's load becomes 30 packets. However, node 10 may have a single parent candidate, say node 20, with a very low remaining energy. Therefore, node 10 has to select node 20 as its parent (since it has no other option) and node 20 has to forward 30 packets, if LBTCA is used, until sink changes its position.

In CBTCA, we improve LBTCA via adding another iteration for checking the conditions when unfair packet load occurs in upper levels of the tree. We check the load and energy level of the ancestors (up to the sink) for the nodes where more than one parent candidate exists. For each of these nodes, we project the remaining energy values of the ancestors of each candidate parent as if they forward the packet load of this node. We find the minimum remaining energy value node of the path for each candidate parent and select the maximum one among them. If this value is in the path of the previously selected parent, then no action is required. Otherwise, we transfer the load of the node to new parent (and to its ancestors). We first remove the load from old parent and all its ancestors (following the path to the sink). Then, the node's new parent is set and this load is added to the new parent and all its ancestors.

Our CTCA algorithm has $O(nH)$ time complexity, where n is the number of nodes, and H is the maximum logical level in the tree (which is given by Equation 4.1). Since all nodes should send their levels and parent ID lists to the sink, total storage complexity becomes $O(n^2(log\, n + log\, H))$.

4.1.3 Distributed Topology Construction Algorithm

Although centralized topology construction algorithm balances the load among the nodes as much as possible, its main disadvantage is delay. After broadcast phase, level and parent list information are sent back to the sink and computations are done by the sink to finalize child-parent relationships. These operations cause a significant delay which can constitute a major problem for delay intolerant applications (like forest and enemy force detection). It is also another issue when

Algorithm 7 Centralized Tree Topology Construction Algorithm

Input: Nodes parent list and hop level information
Output: Parent list and packet load values
procedure CTCA
    $Broadcast(H, pl)$
    for $i \leftarrow 1, n$ **do**
        $Convergecast(H_i, pl_i)$ ▷ ecah node sends H and pl values to sink
        $l_p(i) \leftarrow 1$ ▷ initialize packet load for each node
    end for
    //phase 1: Balancing packet load
    $LBTCA(H, pl, pr, l_p)$
    //phase 2: Transferring loads w.r.t. remaining energy values
    for $lk \leftarrow ml, 1$ **do** ▷ start from leaves
        $cnl \leftarrow find(H, k)$ ▷ find nodes have level of k
        for $i \leftarrow 1, size(cnl)$ **do**
            for $j \leftarrow 1, size(A_{cnl_i})$ **do** ▷ For each ancestor of cnl_i
                $e_{A_j} \leftarrow e_{A_j} - En(pl(j))$ ▷ projected remaining energy value
            end for
            $mv(i) \leftarrow min(e_A)$ ▷ get min energy ancestor
        end for
        $mmv \leftarrow max(mv)$ ▷ get max of min
        if $mmv! = pr(cnl_i)$ **then** ▷ if parent changes
            for $k \leftarrow 1, size(A_{pr(cnl_i)})$ **do** ▷ Decrease load of old ancestors
                $l_p(pr(k)) -= l_p(i)$
            end for
            $pr(cnl_i) \leftarrow mmv$ ▷ update current parent
            for $k \leftarrow 1, size(A_{mmv})$ **do** ▷ Increase load of new ancestors
                $pckLoad(pr(i)) += pckLoad(i)$
            end for
        end if
    end for
end procedure

Node	→	{Parent List}
1	→	{12,15}
2	→	{12}
3	→	{8, 12, 16, 20}
4	→	{7,10}
5	→	{8, 12, 16}

Table 4.1: Sample parent list

sinks are highly mobile and nodes do not have sufficiently large buffer to store data while topology is finalized. We also propose a fully distributed topology construction algorithm in which each node selects its parent individually (without any other message exchange).

When broadcast phase is completed, nodes with the maximum hop-count initiate topology construction process. These nodes can identify themselves from the broadcast phase termination conditions (either H or $2B$ time). In the construction phase, we give a simple heuristic that enables each node to select parents which can evenly distribute load according to the current remaining energy values without excessive message exchange between the nodes.

Since remaining energy value information is included in broadcast packets, a node can select the one with more energy as the parent among several candidates. However, if all nodes use the remaining energy value as the only criteria, then all nodes neighboring to the parent candidate with the largest remaining energy are going to select it as the parent, which can cause an unbalanced topology. We can see this problem with a sample topology given in Table 4.1. Each node lists the candidate parent IDs which it receives packets in broadcast phase. Node 12 has the highest energy value, followed by 16 and 8. If each node selects the candidate with most energy, then nodes 1,2,3, and 5 are going to select the same node, 12, as the parent, which causes unbalanced load distribution. However, node 3 has more options, with the second best alternative, 16.

Our idea is to select parent with probabilities directly proportional to the remaining energy values. We also manipulate the probabilities such that nodes with higher loads are likely to select the parents with more remaining energy.

For that we multiply the maximum remaining-energy value with a factor of l_p/p_t to increase the probability of selecting most advantageous parent for nodes having packet load that is greater than or equal to a packet threshold value (p_t). Multiplying the energy value with a factor more than 1 causes its probability to increase. For instance, assume that candidate parent nodes 12, 16, 8, and 20 has 100, 40, 35, and 25 units of remaining energy, respectively. If node 3 has 15 packets to forward, then it multiplies node 12's remaining energy value (since it is the maximum) by 3, if p_t is 5. Then, node 3 selects those parents with probability 0.75, 0.10, 0.09, and 0.06. This operation increases node 12's probability from 0.5 to 0.75. Since nodes do not communicate with each other during topology construction process, this tuning causes nodes to modify probability values to select the best option without using global network parameters. DTCA's pseudo-code is given in Algorithm 8.

Algorithm 8 Distributed Topology Construction Algorithm (DTCA)

```
1:  procedure DTCA
2:      Broadcast(H, pl)
3:      for i ← 1, n do
4:          mv ← max(e_i)                          ▷ get maximum energy parent
5:          e_i(mv) ← e_i(mv) * (l_p/p_t)          ▷ updates max energy
6:          sum ← Σ_{k=1}^{pl} e_k                 ▷ summation of all parent energy values
7:          for j ← 1, size(pl_i) do
8:              pr(j) ← e_j/sum
9:          end for
10:         ind ← rand(prb)                        ▷ selects one parent w.r.t. prob. values
11:         pr(i) ← pl_i(ind);
12:     end for
13: end procedure
```

In our DTCA algorithm, a broadcast packet has $O(log\, n + log\, H + log\, e)$ storage complexity, where n is the number of nodes, H is the maximum hop-count value, and e is the remaining energy value. For 1000 nodes, 20 hop counts, and 1 joules energy values, it consists of 2 bytes. Each node should keep $O(cp\,(log\, n + log\, e))$ bytes and use $O(cp)$ operations for selecting its parent, where cp is the number of candidate parents.

4.2 Adaptive Load Based Sink Mobility Algorithms for WSN

In the previous chapter, we proposed two different single-sink mobility algorithms that use traffic flow information (either number of packets each node sends, or energy value required to forward other nodes' packets) when sink sojourns at one of the migration points. Most of the sink-mobility algorithms in the literature, like [21,22,27], use this information in their mathematical programming formulations or in their algorithms to decide sink movements in an intelligent manner.

Let x_{ijk} denote the number of packets a node i sends to a node j when sink stays at a point k. Under perfect conditions, x_{ijk} will always be the same (static) for the same i, j, and k values, since we assume the same routing tree will be established for the same i, j, and k values. Using these static x_{ijk} values has a few drawbacks when improving network lifetime. First, it requires an additional training phase to set those values before network starts its operation. This causes additional initial delay. Second, it cannot be possible that those values will be valid all time, those due to reasons like obstacles appearing between nodes, node hardware or software failures, etc. These can decrease lifetime performance, since x_{ijk} values cannot be used as calculated initially in the model. Last, since x_{ijk} values are determined at the beginning and not changed during the lifetime, we could not update them and set energy-efficient topology trees according to the up-to-date remaining energy values of the nodes.

If we can update x_{ijk} values dynamically using up-to-date remaining energy map of the network in our algorithm, constructing energy-efficient trees will contribute to the lifetime more than the algorithms which use static x_{ijk} values. However, using different x_{ijk} values for each round in a mathematical formulation will introduce numerous additional decision variables that will increase time-complexity dramatically. Therefore, a heuristic algorithm can be a good alternative at this point to realize this approach.

4.2.1 Algorithm Details

In this part, we describe our Adaptive Energy-Load–Based Movement Algorithm (A-ELMA) to coordinate sink movement for improving network lifetime. As explained before, A-ELMA does not require any previous visit of sink to the sites. Instead, it starts with an empty matrix T. When network starts its operation, sink selects the first site randomly (say sink site i), goes to this point and starts a topology construction process, as explained above. While the routing tree topology is constructed, each node sends an acknowledgement to the broadcast packet sent by the parent to associate itself with the parent. Since each parent knows its number of associated children, it can calculate its energy load and piggyback this information in the very first packet of a round that is sent to the sink node. The sink node then uses these load values to compute a row i of the matrix T corresponding to the current sink position i. Each element j of the row corresponds to a sensor node j in the network and includes its energy load information, denoted as t_{ij}.

When the time to move again comes, the sink node generates a random value rv (where $0 < rv < 1$) and uses it to make a decision to select another random sink site or select one of the sites that are visited before. If the generated random value is greater than the ratio of the number of visited sites to all sink sites, then the sink goes to a site which is not previously visited, randomly. If not, then it uses the current matrix (non-zero rows which correspond to visited sites) to decide next site using this partial matrix.

Sensor nodes periodically share their residual energies with their parents to be forwarded to the sink. Sink stores this information as a row, in which a column j corresponds to the current residual energy of node j. When sink makes a decision, it subtracts each row of energy expenditure matrix T, from residual energy row e, to estimate the residual energy values of the nodes when sink moves to location i. Sink determines the minimum estimated residual energy value for each sink site location. Then sink finds the maximum value among them and selects the site associated with this value as the next location to move.

As explained above, sink either chooses sites randomly (especially in the beginning) or using the information it collects each time it visits a site. In the first rounds of the network operation, sink usually selects the site randomly to complete the energy-load matrix. Sink uses the matrix more as round progresses and no longer selects randomly after all sites are visited. However, sink will continue to update the values of the matrix T, each time it visits a location for constructing energy-efficient topology trees, which is the main motivation of the algorithm. The pseudo-code of our algorithm is given in Algorithm 9.

In the A-ELMA algorithm, sink stores energy expenditure matrix T and residual energy row e to select sites to move. This corresponds to $O((p+1)\,n\,log\,E)$ storage complexity, where p is the cardinality of the candidate migration point set, n is the number of nodes, and E is the initial energy value of a node (assumed to be the same for all nodes). For 50 sink sites, 400 nodes, and 10 joules of initial energy values, it will require approximately 10 KB for the sink to store this information. The algorithm has $O(p)$ complexity to determine the sink's next site in each round.

4.3 Simulation Results

In this section, we evaluate the performance of our algorithms. First, we compare the performance of the topology construction algorithms. In these experiments, sink uses the same mobility algorithm (A-ELMA), but different topology construction mechanisms are used when sink visits sites. Then we compare A-ELMA algorithm with the following algorithms:

- ELMA: Sink selects sites using Algorithm 2.
- OPT: Number of rounds (x_i) at different sink sites is determined using the mathematical model described in Section 3.2.3.
- RAND: Sink selects sites randomly without considering any network parameter or condition.

Algorithm 9 Adaptive Energy-Load–Based Movement Algorithm (A-ELMA)

procedure A-ELMA(p, t_x) ▷ p: migration points, t_x: transmission range
  $v \leftarrow 0$ ▷ v: # of visited sites
  $vis \leftarrow NULL$ ▷ vis: set of visited sites
  $unvis \leftarrow p$ ▷ $unvis$: set of unvisited sites
  $r \leftarrow rand(unvis)$ ▷ first select a site random
  for $i \leftarrow 1, n$ **do** ▷ for each node
    $T_{ri} \leftarrow tx_i + rc_i$ ▷ initialize row r of T
  end for
  $vis \leftarrow vis \cup r$
  $v \leftarrow v + 1$
  $unvis \leftarrow unvis \setminus r$
  while $e > 0$ **do** ▷ any node's energy is not depleted
    $rv \leftarrow rand()$ ▷ rv: random value, where $0 < rv < 1$
    if $rv > v/p$ **then** ▷ new site will be selected not visited before
      $r \leftarrow rand(unvis)$ ▷ selects a site randomly from unvisited set
      $vis \leftarrow vis \cup r$
      $v \leftarrow v + 1$
      $unvis \leftarrow unvis \setminus r$
    else ▷ previously visited site will be selected
      for $i \leftarrow 1, size(v)$ **do**
        $U_i \leftarrow e - T_i$ ▷ update current energy matrix
      end for
      for $i \leftarrow 1, v$ **do**
        $mp(i) \leftarrow max(U_i)$ ▷ get maximum element for each row
      end for
      $r \leftarrow min(mp)$ ▷ index of minimum of maximums
    end if
    for $j \leftarrow 1, n$ **do** ▷ for each node
      $t_{ij} \leftarrow tx_i + rc_i$ ▷ update row r of T
    end for
  end while
end procedure

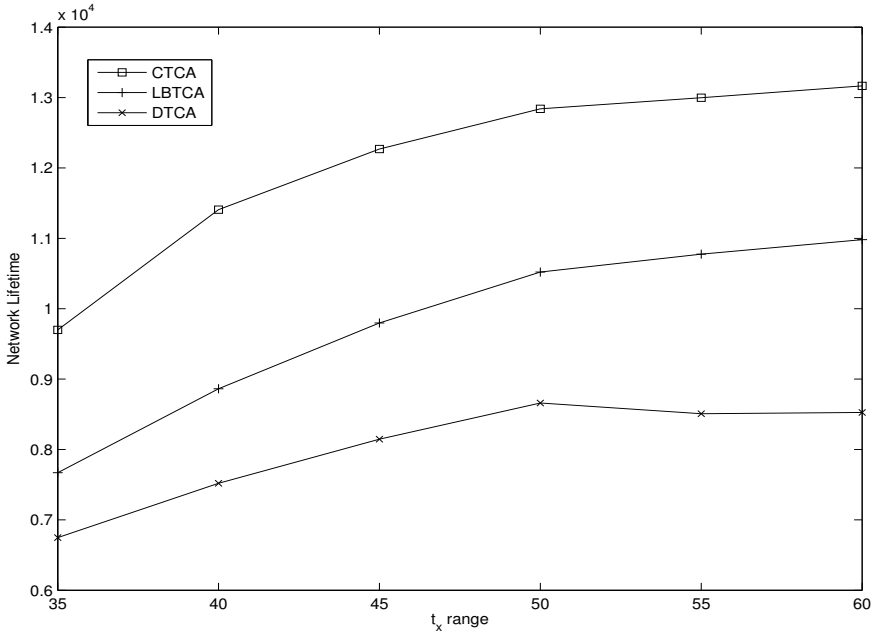

Figure 4.2: Network lifetime versus transmission ranges ($t_{min} = 50$ rounds).

- STS: Sink selects sites at the beginning and stays there.

Topologies are constructed using Algorithm 8 if not stated otherwise. We compared the performance of these five schemes in terms of network lifetime, latency, and total distance sink travels.

Network lifetime values of alternative routing-tree construction algorithms for various transmission ranges is given in Figure 4.2. For all these different alternatives, A-ELMA algorithm is used as the sink mobility algorithm. Number of nodes are fixed to 400 and minimum number of rounds (t_{min}) for each stay of the sink is set to 50 rounds. Results show that CTCA performs better than other approaches. It improves the performance of LBTA by 24% on the average and 50% better than DTCA.

Nodes should send their logical-level and parent-list values to the sink in CTCA and LBTCA algorithms. Therefore, the message transfer cost is higher for these algorithms than that of DTCA. When sink is highly mobile, this cost can affect the

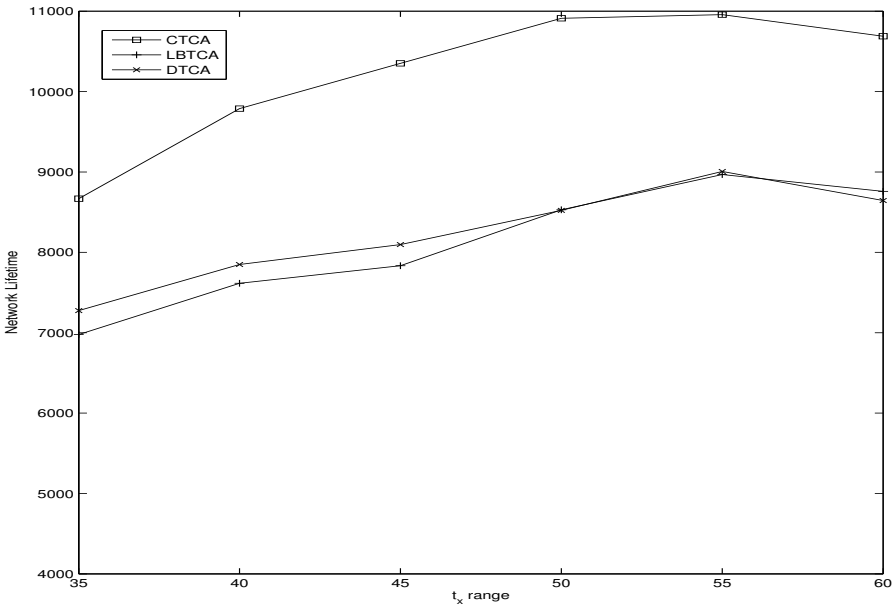

Figure 4.3: Network lifetime versus transmission ranges ($t_{min} = 5$ rounds).

performance of topology construction algorithm. We investigate the performance of these algorithms when t_{min} decreases from 50 to 5 in Figure 4.3. The gap between LBTCA and DTCA (21% on the average) closes when t_{min} decreases. The difference between CTCA and DTCA also decreases from 50% to 24%.

Latency, which is measured in terms of average hop-count to the sink, is another important metric for evaluating the performance of a routing-tree construction algorithm. This metric is investigated in Figure 4.4. The results show that all algorithms have almost same latency (differs from each other by less than 1%).

Network lifetime results for the number of nodes between 400 and 800 are given in Figure 4.5. DTCA is used as routing tree construction algorithm. Lifetime values are decreasing for all schemes when number of nodes is increasing. A-ELMA achieves better lifetime values, 14%, 17%, and 80% compared to OPT, ELMA, and RAND approaches, respectively. Difference between lifetime values increases proportionally with the increase in number of nodes.

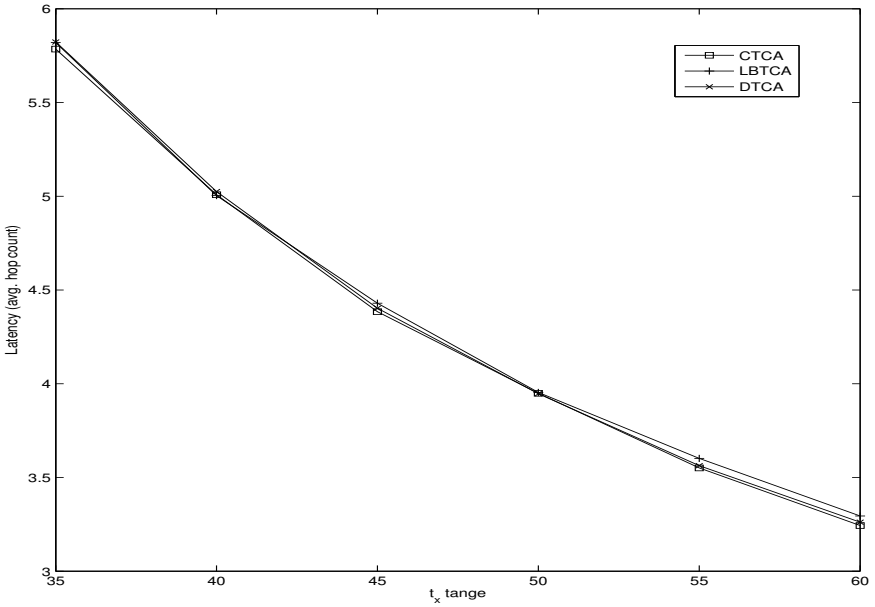

Figure 4.4: Latency versus transmission ranges.

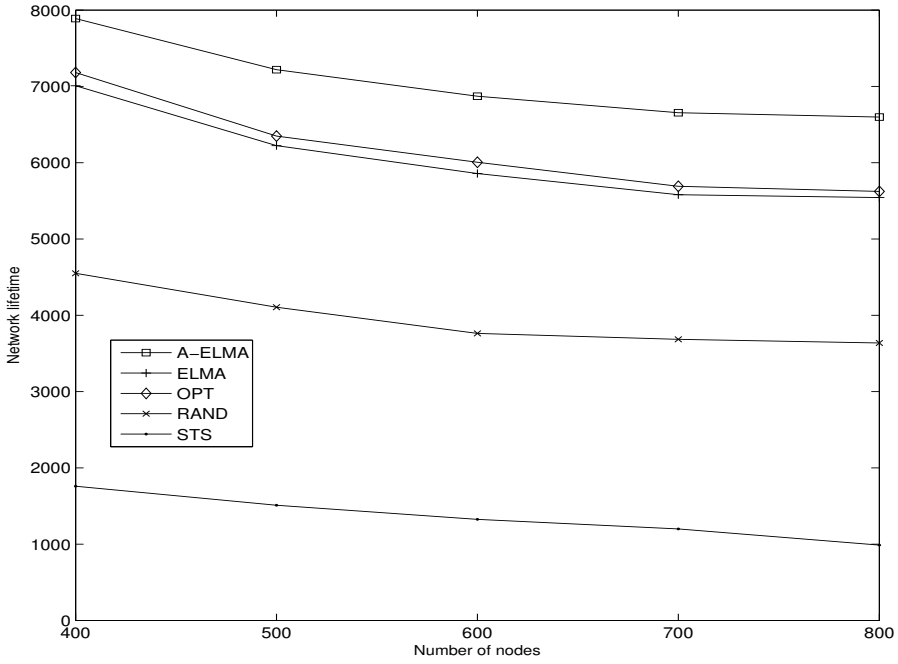

Figure 4.5: Network lifetime versus number of nodes (DTCA).

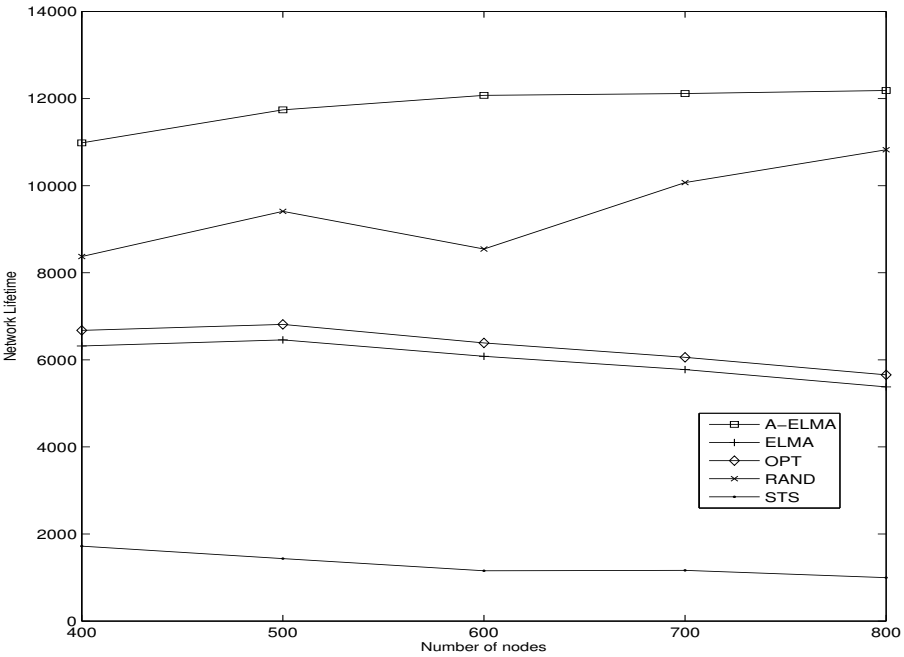

Figure 4.6: Network lifetime versus number of nodes (CTCA).

Network lifetime performance of the schemes when CTCA is used as routing tree construction algorithm is given in Figure 4.6. Lifetime values of the all schemes improve and the gap between A-ELMA and OPT as well as ELMA increases when CTCA is used. A-ELMA has 98% and 88% longer lifetime on the average than ELMA and OPT, respectively. This ratio increases to more than 2, when number of nodes becomes 800. Performance of RAND scheme improves when CTCA is used. It performs 58% and 51% better than ELMA and OPT, but 26% worse than A-ELMA. Lifetime values slightly increase (approximately 10%) for A-ELMA and RAND, unlike the case when DTCA is used, when number of nodes increases. CTCA can efficiently distribute the load than DTCA, since there are more options to select, when number of nodes increases.

We present latency (average hop-count values) for various numbers of nodes in Figure 4.7 (DTCA is used as routing tree construction algorithm). STS has the best (lowest) latency values (24% lower than the others). A-ELMA, ELMA and OPT have almost the same latency values. Their average hop-count values

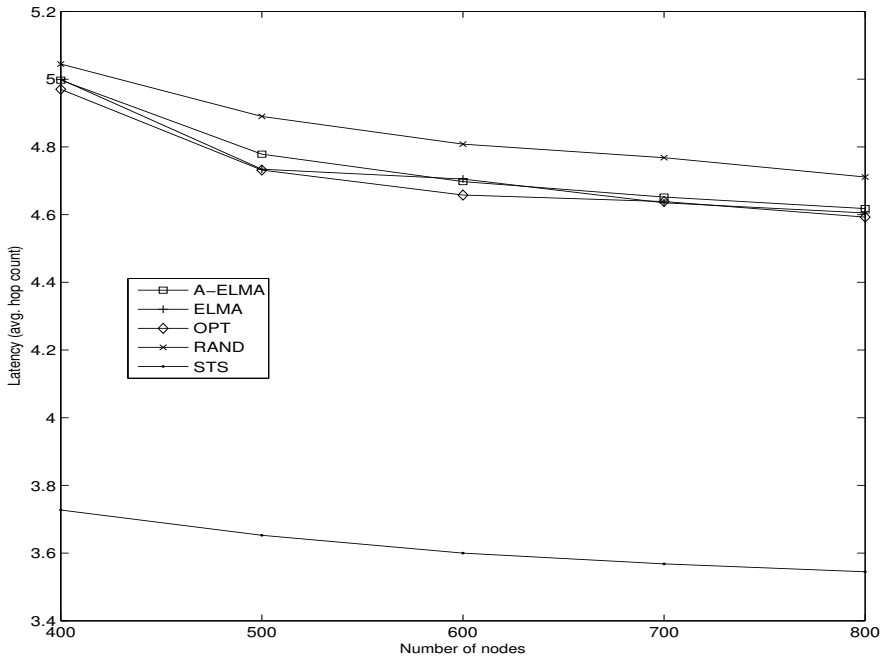

Figure 4.7: Latency versus number of nodes.

are better than RAND by around 2%. The results show the almost same characteristics when CTCA is used as routing tree construction. The only difference is the increase (7% on the average) in latency values for A-ELMA.

Total distance traveled by the sink during network lifetime is given in Figure 4.8. Sink travels the longest distance when A-ELMA algorithm is used, which is 10% and 69% more than ELMA and RAND schemes, respectively. Distance traveled by the sink is directly proportional to the lifetime it achieves. A-ELMA has the best lifetime values than all other schemes; however, sink travels most when using this algorithm to select sink sites. Moving to the different (far from each other) points in the area causes the load to be distributed evenly, which increases the lifetime. Sink travels more than ELMA when it uses A-ELMA or RAND scheme, if CTCA is used. The ratio increases to almost 2, which is equal to the ratio for network lifetime improvement, when number of nodes is 800.

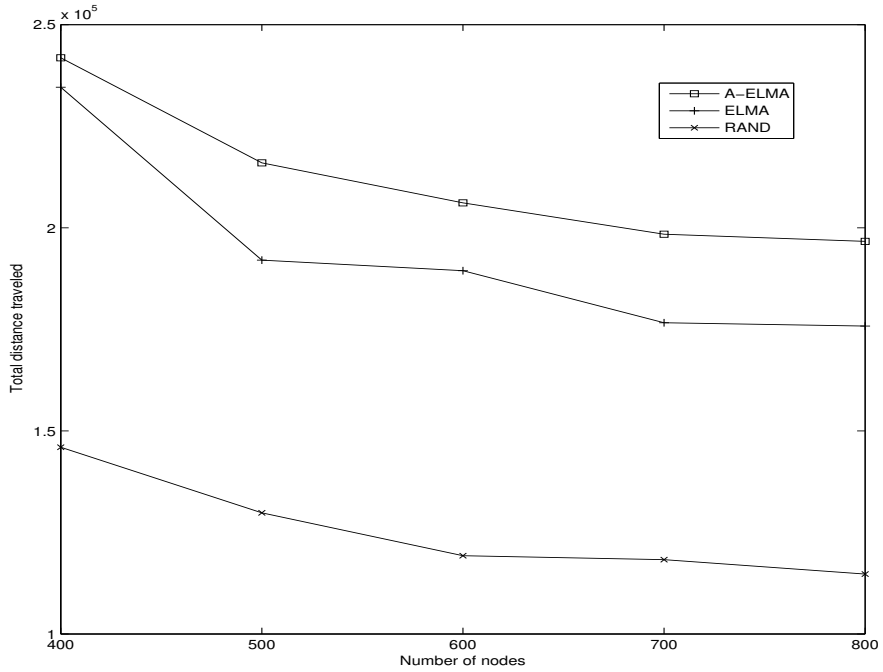

Figure 4.8: Distance traveled for various number of nodes.

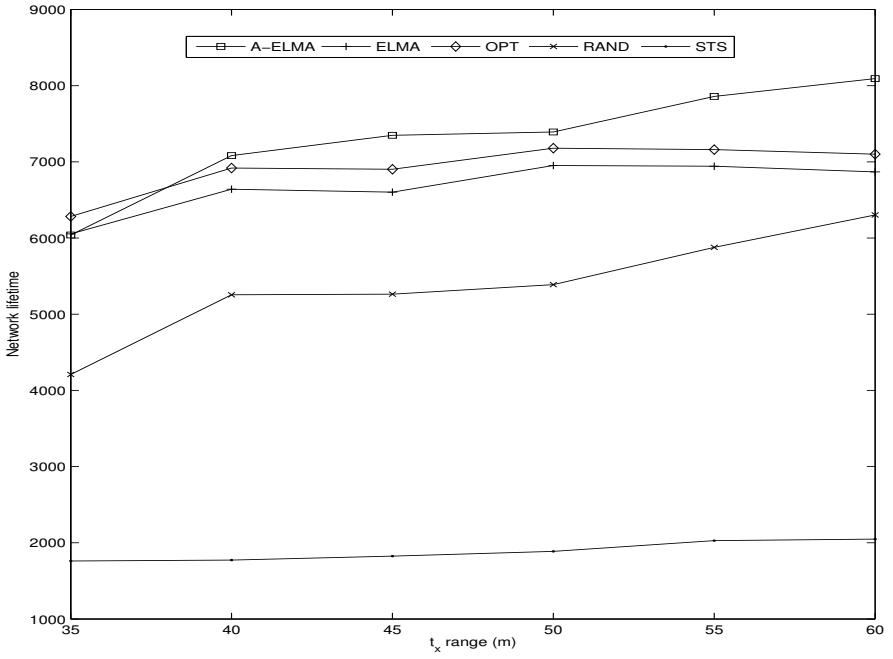

Figure 4.9: Network lifetime versus transmission ranges (DTCA).

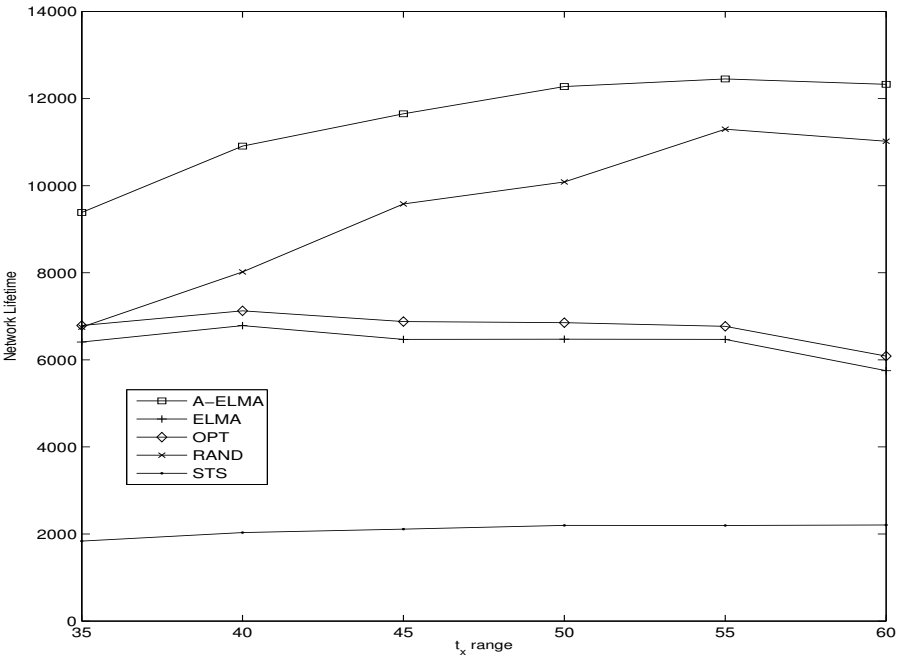

Figure 4.10: Network lifetime versus transmission ranges (CTCA).

The effect of transmission range in network lifetime is investigated in Figure 4.9. Lifetime value increases when transmission range increases since hop count to the sink decreases. Although A-ELMA's lifetime value is lower than OPT when transmission range is 35, it closes the gap and outperforms OPT when range increases (14% better when range is 60). A-ELMA's performance benefits from energy efficient topology construction algorithm, since it uses remaining energy values to construct more load-balanced trees.

The effect of transmission range when CTCA is used is given in Figure 4.10. A-ELMA performs better than ELMA and OPT schemes and the difference between the results increases (from 40 % to 100%) when transmission range increases. A-ELMA also performs better than RAND; however, this ratio decreases when transmission range increases. Nodes will have more parent candidates when transmission range increases and this situation enables the CTCA to uniformly distribute the load. This increases the effect of routing tree construction algorithm (CTCA) in improving network lifetime compared to mobility scheme.

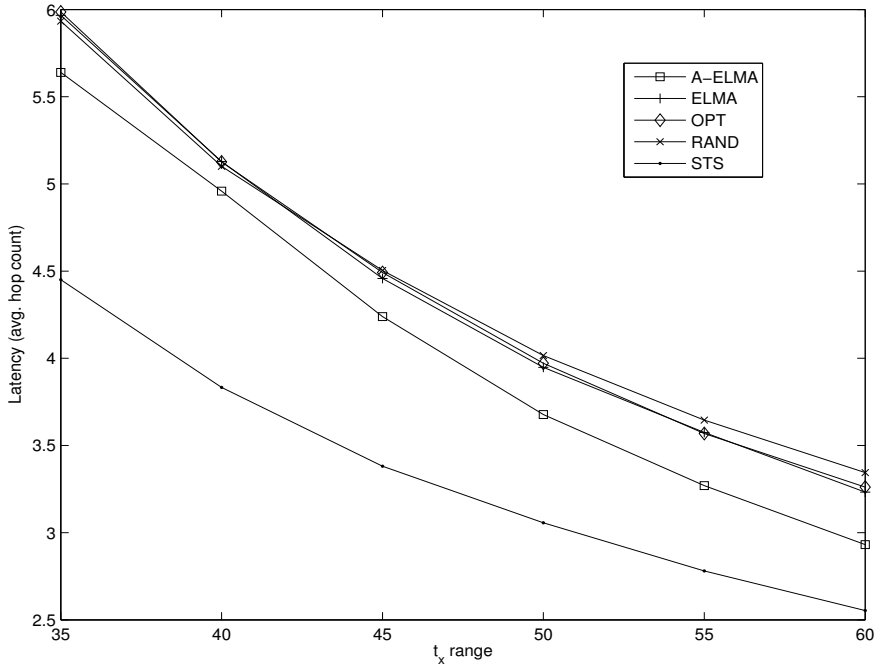

Figure 4.11: Latency versus transmission ranges.

Latency values for transmission range values between 35 and 60 are given in Figure 4.11. ELMA, OPT, and RAND have almost same latencies. A-ELMA has better (lower) latency (10% on the average) than these three schemes. However, STS has the lowest latency values since the aim is decreasing the nodes' average hop-count to the sink when placing the sink in the area.

Sink can stay a minimum amount of time (fixed number of rounds, called t_{min}) after moving a point when topology construction or mobility cost is high. We investigate the effect of this factor, t_{min}, to the network lifetime in Figure 4.14. A-ELMA's performance decreases dramatically (82% lower when t_{min} increases from 50 to 250) when sink tends to stay longer at a site. Since, A-ELMA does not have any prior information about the network; it needs to visit all sites to complete its matrix. When this operation is delayed, its performance is affected more. This is not the case for OPT and ELMA, because they start with a full matrix.

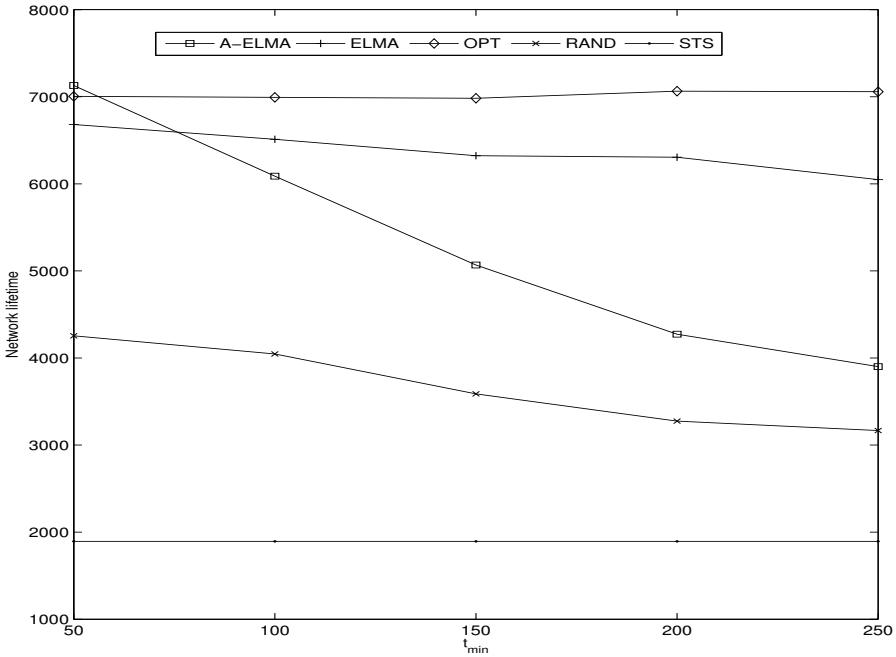

Figure 4.12: Network lifetime versus t_{min} values (DTCA).

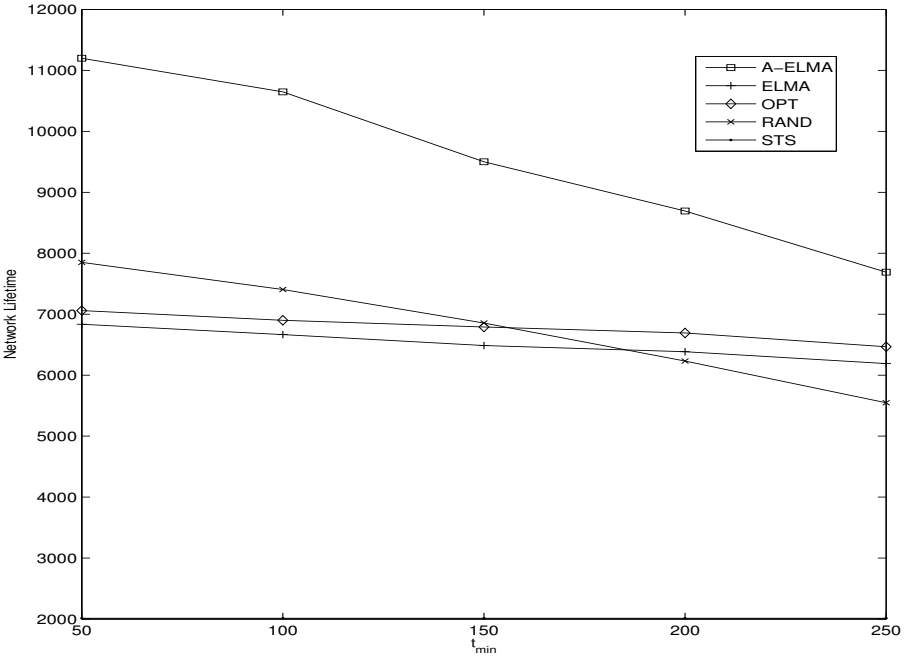

Figure 4.13: Network lifetime versus t_{min} values (CTCA).

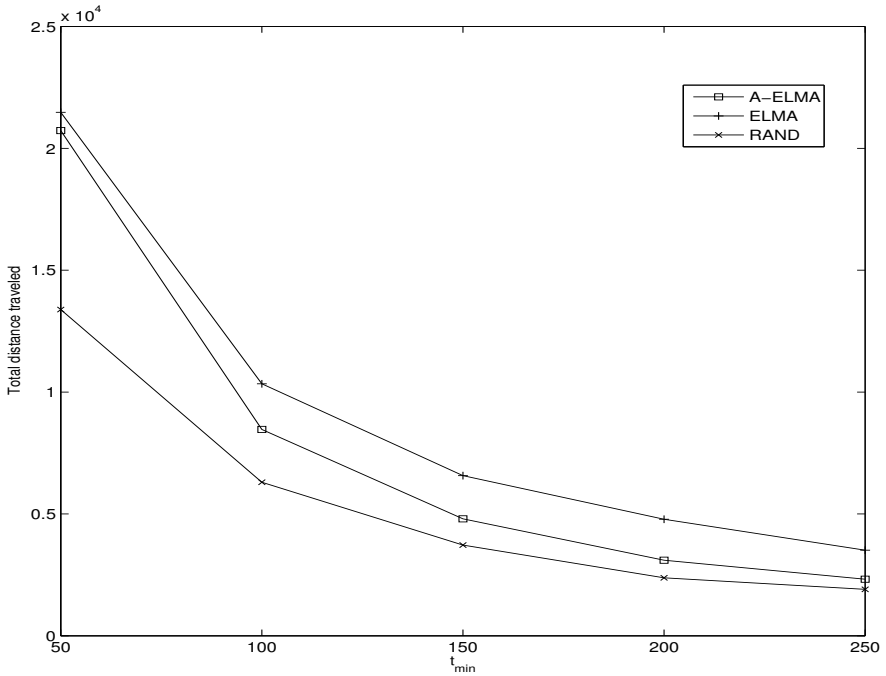

Figure 4.14: Distance traveled for various t_{min} values.

The effect of t_{min} value when CTCA is used in network lifetime is investigated in Figure 4.13. Although the difference between A-ELMA and other schemes decreases when t_{min} increases, it can still perform better for large t_{min} values unlike the case where DTCA is used. Although A-ELMA's performance degrades (around 45%) more than ELMA and OPT (approximately 10%), it is almost half of the 82% in DTCA case. CTCA's contribution to the lifetime partially compensates the performance degradation caused by increased of t_{min} value.

Distance traveled values for various t_{min} values is given in Figure 4.14. The distance traveled values drop by a factor of 8, 7, and 6 for A-ELMA, ELMA, and RAND, respectively. Since ELMA's lifetime performance is not affected so much by increasing t_{min} (decreases only 10% percent when t_{min} increases from 50 to 250), it would be better to stay longer to reduce mobility cost when using ELMA. If we visit a site i for 250 rounds when lifetime finishes, it does not make any difference to visit there three times and stay 100, 50, and 100 rounds or visit once and stay 250 rounds in terms of lifetime. However, it will make difference

in terms of mobility cost and the algorithm can compensate this *long* stay and probably does not choose this site once more, however, still can achieve almost the same lifetime comparing with small t_{min} value. On contrary, A-ELMA needs small t_{min} values to learn network parameters as soon as possible to make smart decisions. It will be a good alternative to use a hybrid approach (small t_{min} values until matrix is fulfilled and increasing it thereafter) to make a balance between lifetime improvement and mobility cost.

4.4 Conclusion

In this chapter, we propose two routing tree construction algorithms and an adaptive sink mobility algorithm to improve network lifetime. Routing tree construction is an important component in sink mobility, since each time sink visits a location, it reconstructs a routing topology. CTCA algorithm improves the performance of LBTA by checking the all nodes' remaining energy values in the path to the sink and the load is transferred if a more advantageous path is formed. In DTCA, nodes select their parents proportional to remaining energy values. However, they do not send excessive messages related to topology construction.

Although we previously propose algorithms for single-sink mobility problem, there are some limitations. They require a training phase to determine child-parent relations and stick to these decisions through the lifetime. However, some conditions can prevent nodes to select same parent each time sink visits the same location. A-ELMA algorithm does not have a training phase and starts with no information about network. Sink learns these values during the lifetime and select sites either randomly (especially at the beginning) or using the current information it learns.

Simulation results show that A-ELMA outperforms ELMA and OPT algorithms in terms of network lifetime. Its performance increases when CTCA is used as routing tree construction algorithm. However, sink travels more when using A-ELMA. Its performance degrades when sink has to stay more (large t_{min}

value), because it could not learn the required parameters earlier to improve the lifetime.

Chapter 5

Coordinated Movement of Multiple Mobile Sinks

In this chapter, we present two low-complexity algorithms for multiple-sink mobility problem, Multiple-Sink Movement Algorithm (MSMA) and Prevent and Move Away (PMA) algorithm. Our algorithms have low complexity and low overhead, and therefore, can be used directly with sensor nodes and mobile sink nodes.

Our MSMA algorithm runs in each sink node and relies on the energy consumption information of each sensor node for a subset of all possible sink-site combinations. Most algorithms in literature use traffic-load information for each possible sink-site combination as part of their solutions. Since the number of sink-site combination alternatives increases exponentially, it becomes impractical to calculate traffic-load for every sink site combination for large number of sites and sinks. In our MSMA algorithm, we address this problem and we do not require enumeration of all possible sink-site combinations. Our algorithm uses a threshold and limits the number of combinations to consider for movement of sink nodes.

Our PMA algorithm is a totally distributed algorithm and does not require

any prior information about the network and possible traffic-load of nodes for specific sink-site combinations. Each sink checks the residual energy values of its descendant nodes and *forbids* placement in sites nearest to these low-energy nodes. This information, as well as the minimum energy nodes available within the sites' first-hop neighbors, are shared among sinks. Then, half of the sites are selected according to these values. Other half of the sites are chosen to be far away from the previously selected sites so that sink nodes are not clustered to a particular sub-region in the area and the load is balanced as much as possible without collecting and using global network information.

The rest of the chapter is organized as follows: We describe the system model we used in Section 5.1. We present our proposed algorithms in Section 5.2, and give the performance evaluation of them in Section 5.3. Finally, we give our conclusions in Section 5.4.

5.1 System Model

We consider a sensor network with n stationary sensor nodes deployed to an area of interest. The set of sensor nodes is denoted with $|N| = n$. There are s mobile sink nodes in the environment, which may change their locations periodically according to network conditions. Sinks choose these locations from a predetermined sojourn point set P, where $|P| = p$. Each sensor $i \in N$ has a packet generation rate of Q packets per second.

Tree-based routing is used such that each sensor node sends its own packet and relays its children's packets to the same parent for a given sink-site combination. Sinks learn the related parameters in a *training* phase so that the same routing tree is constructed in each visit to the same site combination. This method enables us to construct an energy-load matrix structure T of sink-site combinations (rows) and nodes (columns), such that each entry t_{ij} in the matrix lists how much energy a node j would consume to transmit its packets to its parent in the routing tree for a given sink-site combination i.

Sink sites (sojourn points) are either predetermined or chosen using a sink-site determination method. Some example methods are given in [72]. Those sink sites are associated with each sink using a method that we explain in the next section. Sinks visit those sites (combinations) together and start the topology tree formation process to construct their matrices if sensor nodes' positions are not known a priori.

In this phase, each sink broadcasts a packet to construct a tree rooted at that sink using a tree-topology construction algorithm. When the tree has been set up, each sensor node saves the topology ID and parent ID pair to remember which node to send to when sinks select the same site combination at any time over the network lifetime.

When a node receives a packet from a different sink, it saves its ID and hop-count to the sink even if this hop-count is bigger than its current value. It becomes a *junction* node and it informs all the sinks it receives packets from. These junction nodes can be used for two different purposes: sink communication and topology maintenance.

If junction nodes are known by the sinks, then those sinks can reach each other using the paths through junction nodes. Although sinks have higher transmission power than sensor nodes do, sometimes they can be far enough from each other that direct communication cannot be achieved. In these cases, using those junction nodes help sinks exchange information for next site selection decision or for remaining energy values.

In delay-intolerant applications, it is always critical for sensor nodes to send their data to an available sink (anycast) instead of buffering it. Multiple sinks do not move just exactly at the same time since at least one of the sinks must stay to gather sensor nodes' data. When a sink decides to move, it informs its neighbor nodes and sends a re-connect request packet to this junction node indicating that the junction node should connect to another network. This packet is sent using the reverse path from the junction node to the root. Sink's first-hop neighbors change their parents toward the junction node. For the subtree that

junction node belongs, its ancestor, which is one of the sink's first-hop neighbors, reverses the packets' direction. For other subtrees, rooted at the sink's other first-hop neighbors, these nodes forward their packets to the junction node's ancestor, until a new topology construction packet with smaller hop-count value is received by the nodes, which means a sink arrives its new location. There are some works about mobile dynamic tree construction and maintenance, such as [79–81], however this topic is beyond the scope of this chapter.

5.2 Proposed Algorithms

As mentioned in Chapter 2, most studies about multiple-sink mobility in the literature use linear/integer programming formulations to achieve optimal values of network lifetime. These methods, however, cannot be used by sink nodes on-the-fly during network operation, since they require huge amount of computation which sink nodes cannot achieve with the current technology. In this chapter, we want to present algorithms that do not require too much complexity; but still achieve considerable lifetime improvement compared to random movement and static-sink cases.

5.2.1 Multiple-Sink Movement Algorithm (MSMA)

Our first algorithm to coordinate the movement of multiple mobile sinks for improving network lifetime is called MSMA. The algorithm extends our earlier algorithm MS-ELMA [82]. We improve the method of limiting the possible position combinations before the network begins operation. We also change the algorithm to add limited combinations on-the-fly to further improve its performance.

Algorithms proposed to solve the multiple-sink mobility problem usually consider the amount of traffic (directly or indirectly) between nodes when sinks are placed at certain locations/sites [27,68]. Assume each different sink placement is a sink-site combination. A set of sinks has many possibilities to get placed into

these locations. When sinks are placed according to a combination k and routing trees are formed from sinks to nodes, the traffic from a node i to its parent node j can be denoted with x_{ijk}. Knowing x_{ijk} values for all possible site combinations as well as the current remaining energy values of nodes help us to select the best alternative site combination to move, because we can see how the nodes' energy values will be affected if sinks would move to a specific site combination. However, calculating x_{ijk} values for all possible site combinations before network starts its operation can take a very long time, which is not feasible for reasonable p and s values, such as 50 sink sites and 7 sinks.

Since choosing s points among p alternatives increases exponentially, the main idea should be to decrease those alternatives as much as possible to reduce complexity without sacrificing lifetime improvement. For instance, choosing seven locations among 50 alternatives generates approximately 10^8 alternatives! In our algorithm, we require all sinks to make a decision about their next migration points using their information, which are further evaluated and a final decision is made collectively.

Each sink uses an energy-expenditure matrix in which each element t_{ij} corresponds to the energy required at a node j to transmit packets to its parent when the sink nodes are in position-combination i, i.e., the combination tuple - $(p_1, p_2, \ldots p_s)$. The number of columns of this matrix apparently equals the number of nodes n in the network. The number of rows, i.e., the number of position combinations, is, however, a parameter in our algorithm, which can be determined using sink characteristics. This is called *binomial threshold* and is not directly involved in our algorithms, instead affects the th_s parameter (threshold for each sink), which determines how many migration points each sink will select to calculate different combinations. In other words, th_s is the maximum integer satisfying $\binom{th_s}{s} \leq t_b$, where s is the number of sinks and t_b is the binomial threshold. If we select t_b as 2000, then th_s becomes 13, when the number of sinks is eight, since $\binom{13}{8} = 1287$.

After determining these parameters, we select the possible migration points for each sink. There will be a total of s groups of points. For a sink, we assign

possible migration points from different regions of the area. First, we determine the number of points (k) to assign for a sink. It is given by $k = p/s$. Then, we start using the *k-means* algorithm to select migration points for sinks. For the first sink, we run the k-means algorithm considering the whole migration point set. Given the whole point set (p points initially), the k-means algorithm selects k centroids (positions) and the k migration points with minimum distance to these k centroids are assigned to the first sink. We remove these selected positions from our point set and we continue to select points for the next sink by re-running the k-means algorithm to select another k centroids from the remaining point set. This process continues until all points are partitioned among the sinks. If allocated number of points is smaller than the predetermined threshold value, then new points are assigned randomly to close this gap.

After each sink has its group of sites, i.e., possible position-combinations to move, it will construct an energy-load matrix. For each candidate sink point i, that means for each row, the algorithm computes how much energy the nodes would consume to transmit their packets to the next node (parent) in the routing tree for that sink position i. If coordinates of the nodes are known apriori, these matrices can be constructed by a central authority, i.e. via one of the sinks. The other option is to visit those location combinations and form topology trees for each combination to determine the child-parent relationship and energy expenditure values to construct matrices in a training phase, as in [68], before the network starts operation. Each sink keeps its own matrix that has $\binom{th_s}{s}$ rows and that includes combinations of the points previously assigned to it. Sinks use these matrices to take role in point selection process. Combination with best value is selected for next migration point collectively. Therefore, sinks are not limited with the points in their matrices, instead can go to any point in the area according to the solution given by the algorithm.

Sinks use nodes' remaining energy values and the energy-load matrix to determine the next migration points. The sensor nodes' remaining energy values, e, can be represented as a 1xn matrix (a vector or a row). Sinks share their descendants' values with each other to make the row complete for all entities.

Each matrix row indicating a sink-site combination is subtracted from this remaining energy row to project the nodes' remaining energy values if sinks moved to those locations in the next round. The sinks focus on the minimum values of each subtracted row and select the maximum of them. These values, i.e., the maximum energy values and sink-site combinations, are sent to the sink with the minimum-ID through either single-hop or multi-hop communication. If multi-hop communication is needed, then junction nodes are utilized. A sink compares the received values with their own and sends the one with the maximum value to the other sinks. Last, the decisive sink compares all received values and picks the maximum. Then, the selected migration point information is sent back to the sinks in the reverse direction.

In the MSMA, sinks can also select migration points randomly but with a small probability ($s/100$). Since the algorithm starts with fewer combinations, it is a good idea to increase these combinations without exceeding the t_b value. But more important, since the topology is constructed at the selection, the algorithm should take advantage of the sensor nodes' current remaining energy information. Instead of using the topology constructed using full energy levels (prior to the network beginning operation), sinks can construct more energy-efficient topology trees using the nodes' current residual energy. Since sinks use this option rarely, selecting those sites randomly is a good choice because it decreases decision time and message overhead.

Although the next migration points have been determined, which sink is going to exactly which point has not been set. This decision can also made by the same sink, since it knows the current location of each sink. Sinks should inform other sinks about their locations to prevent two or more sinks going to the same location. This is the well-known *assignment problem*, and the Hungarian method can find the optimal solution for assigning the sinks to new locations with $O(n^3)$ complexity [83].

After sinks migrate to the new points, they will operate there until a threshold is reached; either a certain amount of change in the energy of its first-hop neighbors is detected or a fixed number of rounds - t_{min} - has been achieved.

Then, when the sojourn time expires, sinks share their children's remaining energy values with each other and again run the max-min search over the matrix to determine new migration points. This iteration continues until a node depletes its energy. The algorithm is detailed in Algorithm 10. It takes $O(n\,t_b)$ to construct the energy load matrix. In each round, we need $O(t_b)$ computation to determine the sinks' next migration points.

5.2.2 Prevent and Move Away (PMA) Algorithm

In this section we describe our distributed Prevent and Move Away (PMA) Algorithm, which is fully distributed and does not require excessive information exchange. As explained above, the current schemes proposed in the literature consider the amount of traffic passing through each node to its parent while sinks are at specific sites. Although this approach improves network lifetime significantly, considering all possible sink site combinations is computationally expensive. Therefore it is important to also come up with algorithms which do not require evaluation of so many combinations and distributing the related information while selecting next migration points.

Although there are a few distributed algorithms that attempt to solve the multiple-sink mobility problem, most require a large amount of data exchange, e.g., nodal residual energy, nodal data rate, energy needed to communicate packets for the whole network, as in [68]. Transferring such a large amount of data in each round causes excessive energy consumption in resource-constrained sensor nodes. Therefore, it is very important to find a low complexity and efficient heuristic that requires minimum information exchange to decide sink movements and that can be run on sink nodes, which is what we achieve with our PMA algorithm.

Our PMA algorithm first filters out the sites closer to the nodes that have relatively lower residual energy compared to other nodes in the network. We do this because we only want to consider the sites close to high energy nodes as possible migration points so that when sinks migrate to those points, the high

Algorithm 10 Multiple-Sinks Movement Algorithm

procedure MSMA($c, t_x,$) ▷ c: node coordinates, t_x: transmission range
  $p \leftarrow getMigrationPoints(c, t_x)$ ▷ determine possible migration points
  $cs \leftarrow \lfloor size(p)/s \rfloor$ ▷ initial cluster size for each sink
  $p\prime \leftarrow p$
  for $i \leftarrow 1, s$ **do**
    $cntrs \leftarrow kmeans(p\prime, cs)$ ▷ get cluster centers using *kmeans* algo
    $p_i \leftarrow pdist(p\prime, cntrs)$ ▷ migration points nearest to centroids
    $p\prime \leftarrow p \setminus p_i$
    if $cs < th_s$ **then** ▷ if # of centroids < threshold for given # of sinks
      $df \leftarrow th_s - cs$
      $p_i = p_i \cup rand(p_i\prime, df)$ ▷ finalize point set
    end if
    $cmb_i \leftarrow combos(p_i, s)$ ▷ get combinations
  end for
  for $i \leftarrow 1, s$ **do**
    for $j \leftarrow 1, size(cmb_i)$ **do**
      $tt \leftarrow constructTopology(c, t_x, cmb_i(j))$
      for $k \leftarrow 1, n$ **do** ▷ for each node
        $t_{ijk} \leftarrow tx_k + rc_k$ ▷ transmission and receive cost for node k
      end for
    end for
  end for
  $rgVal \leftarrow 1 - (s/100)$
  while $e > 0$ **do** ▷ any node's energy is not depleted
    for $i \leftarrow 1, s$ **do**
      if $rn > rgVal$ **then**
        $sc \leftarrow rand(p, s)$ ▷ select s points randomly from site set p
      else
        $u_i \leftarrow e_i - t_{ijk}$ ▷ update current energy matrix
        for $j \leftarrow 1, size(cmb_i)$ **do**
          $mp(j) \leftarrow min(t_j)$ ▷ get minimum element for each row
        end for
        $r_i \leftarrow max(mp)$ ▷ index of maximum of minimums
      end if
    end for
    $r_m \leftarrow max(r_1, r_2, \ldots, r_s)$ ▷ get global maximum
    for $i \leftarrow 1, s$ **do**
      $sc_i \leftarrow cmb_m(r_m, i)$ ▷ set sinks' next coordinates
    end for
  end while
end procedure

energy nodes will take the majority of the traffic-forwarding load since they will be at the highest level in the routing trees. PMA also considers the distances among the selected next migration points. The selected points should not be very close to each other, since moving all sinks to the same sub-region is not useful and makes the routing paths to the majority of sensor nodes to be excessively long. Therefore, we provide a balance between energy and distance while considering the next migration points. We select half of the sites to be close to high energy nodes and the other half to be far away from these selected sites. In this way, the selected sites are tried to be evenly distributed to the whole sensor network region and are not clustered to a sub-region.

In PMA, the next migration points for sinks are determined as follows at each round of sink site determination. At the end of the current round, before determining next migration points for the next round, the sinks are at some locations and their combined routing trees span the whole network. Each sink has its own routing tree spanning a portion of the nodes in the network. These nodes are considered to be the descendants of that sink. The routing trees of sinks do not overlap.

Each sink knows its descendants and their remaining energy levels (residual energy information is piggybacked into the data packets). Each sink selects the m_p percent of the minimum-energy nodes among its descendants, where m_p is a design parameter. The sink then determines the nearest sites to those minimum energy sensor nodes to filter out for the next round. These filtered out sites are *forbidden*. All sinks then exchange this forbidden site information among each other. This requires a total of $s(s-1)$ message exchanges among sink nodes. Each sink then aggregates the forbidden site information coming from other sinks (takes the union) and in this way obtains a forbidden site list for the whole network. The remaining sites are *allowed* sites and can be used as possible migration points for the next round. Since the forbidden sites can be significant portion of the original point set, we use a parameter called *forbidden point threshold (t_{fp})* to control its size. The total number of forbidden sites cannot be bigger than this threshold. The introduction of this threshold eliminates the possibility of having the number of allowable sink sites to be less than the number of sinks s. Additionally, filtering

out too many sites without global network information (like the energy matrix in the MSMA or MinDiff-RE) can eliminate advantageous sites to move to.

When a sink decides its forbidden sites, that means it decided on its allowable sites. For each allowable site in the region of its routing tree, the sink considers the sensor nodes around the site and finds out the node with minimum remaining energy. This minimum energy level is associated with that site. Then the sink selects the first $s/2$ sites that have the largest associated minimum remaining energy levels. In other words, the first $s/2$ sites are selected when sites are sorted in decreasing order with respect to their associated minimum energy levels. We call these as max-min sites. They are good positions for sinks to move next. Then each sink sends $s/2$ max-min sites together with the corresponding energy values to a designated sink (min id sink, for example). This requires a total of $s-1$ message transfers to the designated sink. In this way, the designated sink collects $s(s-1)/2$ max-min positions. It then selects $s/2$ of them, i.e., the first $s/2$ sites with largest associated minimum energy values.

After this selection, the remaining $s/2$ sites are determined one by one according to the total distance to previously selected sites. First, distance to each of the $s/2$ previously selected sites is calculated to determine total distance value for each remaining *allowable* site. Site with maximum value is selected as the $(s/2)+1^{\text{th}}$ point and added to the previously selected site list. The iteration is done for each site until all remaining $s/2$ sites, that are far away from each other, are determined. This process is sequential since determining the remaining $s/2$ sites once (from the sorted list according to the maximum distance) can cause selecting sites near to each other.

After s sites are selected, s sinks are assigned to these sites using the well-known Hungarian assignment algorithm, like in the MSMA. This minimizes the total distance traveled while sinks move to the assigned sites afterwards. The other alternative is to notify the nearest sink to move as each sink site is determined, if network is delay-intolerant or movement cost is not very significant.

In this way, the algorithm finds a balance between the energy and distance

components of the problem. It takes the energy values of the nodes into consideration in the first part by removing the sites close to low energy nodes. It also increases the distance between sinks to balance the forwarding load of the sensor nodes. The process is described in Algorithm 11. In each round, we need $O(s)$ computations to determine the next migration points.

In the PMA algorithm, each sink shares the site IDs of the forbidden sites (t_{fp} sites for each sink, in the worst case), as well as the $s/2$ minimum energy values (with site IDs) in each movement decision. This corresponds to $O(t_{fp}\, logp)$ and $O(s\,(log\, p + log\, E))$ (where E is the initial energy value in microjoules) storage complexity, respectively. For 50 sink sites, four sinks and 10K microjoules (t_{fp} becomes 10), it requires 60 bits for each sink to share t_{fp} (10, for this case) forbidden site IDs and 40 bits for $s/2$ (2, for this case) minimum energy values and site-id pairs. $s(s-1)$ message exchanges are done sharing forbidden site IDs and $s-1$ exchanges are required for energy and ID pairs to flow across all sinks. If network lifetime is 500 rounds, total message exchange of all sinks during this lifetime becomes around 51 KB.

In the approach described in [68], the information exchange part is much more expensive, where each sink shares all its descendants' residual nodal energy levels with the others. The maximum energy amount that a node can have is divided into 40 levels (L), which requires six bits to represent. Sinks also need to append sensor IDs to a message so each counterpart can construct a full residual-energy-level table. This operation requires additional $log\, N$ bits. Since all nodes' energy levels should be shared, totally $N(log\, N + log\, L)$ bits are needed to convey the required information for a decision. For 500 nodes (and 40 levels), it requires 7500 bits for one sink to send the related information. For 500 rounds and four sinks, the total message exchange cost increases to 5.36 MB for the algorithm of [68]. This is 108 times more than what our PMA algorithm needs.

Algorithm 11 Prevent and Move Away Algorithm

1: **procedure** PMA(c, t_x, cm, s, n) ▷ c: coordinates, t_x: transmission range
2: $cmp \leftarrow cm$ ▷ gets a copy of cm: candidate migration points
3: **while** $e > 0$ **do** ▷ any node's energy is not depleted
4: **for** each round **do**
5: $men \leftarrow (n \times m_p \times s)$ ▷ # of min. energy nodes to consider
6: $t_{fp} \leftarrow (|cm| \times m_p \times s)$ ▷ forbidden site threshold
7: **for** $i \leftarrow 1, s$ **do** ▷ for each sink
8: $min\_nodes(i) \leftarrow min(energy, men)$ ▷ min. energy nodes
9: **end for**
10: **for** $i \leftarrow 1, s$ **do**
11: $cmf(i) \leftarrow dist(cm, min\_nodes(i))$ ▷ get closest sink sites
12: **end for**
13: $cmf \leftarrow cmf_1 \cup cmf_2 \cup \ldots cmf_s$ ▷ forbidden sink sites
14: $cmf' \leftarrow cmf(1 : t_{fp})$ ▷ get first $\#t_{fp}$ sink sites
15: $cmp \leftarrow cmp - cmf'$ ▷ forbidden sink sites
16: **for** $i \leftarrow 1, \lfloor s/2 \rfloor$ **do**
17: $sc_i \leftarrow maxmin(cmp)$ ▷ choose $s/2$ sites with max-min
18: **end for**
19: **for** $i \leftarrow \lfloor s/2 \rfloor + 1, s$ **do**
20: $sc_i \leftarrow max(dist(cm', \sum_{j=1}^{i-1} sc_j))$ ▷ choose furthest sites
21: **end for**
22: **end for**
23: **end while**
24: **end procedure**

5.3 Performance Evaluation

In this section, we discuss the results of our MATLAB-based performance evaluation of the proposed schemes. We compare our MSMA and PMA schemes with four different schemes:

- MinDiff-RE: Azad and Chockalingam's algorithm minimizes the residual energy difference [27].
- MS-ELMA: Our naive algorithm that limits possible site-sink combinations and selects the one that maximizes the minimum remaining energy [82].
- RAND: Sinks select sites randomly without considering any network parameter.
- STS: Sinks select sites at the beginning and stay there.

We compared the performance of these six schemes in terms of network lifetime, latency, and total distance sinks travel.

5.3.1 Simulation Parameters

The sensor networks in our simulations have N static sensor nodes randomly deployed to a region and s mobile sinks (base stations). The deployment region is a square-shaped region with area size 300×300 m^2. After mobile sinks move to their initial locations, they start broadcasting topology construction messages to construct tree-based multi-hop routing topologies (routing trees) from top to bottom. After topology construction, each sink node will have its routing tree constructed rooted at itself. Routing trees do not overlap. We use a constant packet generation rate Q (1 packet/s) for each sensor node and m_p is set to 0.05 (five percent). Simulation parameters and their typical values are summarized in Table 5.1.

Table 5.1: Simulation parameter list

Parameter	Value
Area	$300 \times 300\ m^2$
Number of sensor nodes	500
Node deployment	random and uniform deployment
Transmission range	40m
Data routing	Shortest path
Sink site determination	Neighborhood-based SSD Algorithm [72]
Nodes' initial energy	50 J
Radio characteristics	First-order radio model [77]

5.3.2 Simulation Results

Network lifetime results for the number of sinks (sink count) between two and eight are given in Figure 5.1. Our MSMA algorithm gives a better network lifetime compared to other schemes for all values of sink count. Even MinDiff-RE uses four times more combinations than MSMA when the number of sinks is eight, the MSMA gives a three percent better lifetime performance compared to it. The PMA algorithm performs 22% worse compared to centralized algorithms on the average, but it still outperforms RAND approach by 21% on the average.

Data latency (average hop-count) is another important metric for wireless sensor networks. Algorithms should have lower data latencies, especially for delay-intolerant applications (like fire detection). The latency comparison of the schemes is given in Figure 5.2. The static sink approach has the best (lowest) latency values - 10% lower than the centralized algorithms and 13% and 16% lower than the PMA algorithm and the RAND approach, respectively. Centralized algorithms have almost the same latency as each other. The PMA algorithm also has the same latency as centralized algorithms, but it has lower latency (3% on the average) compared to the RAND.

Although it is not taken into consideration most of the time, travel distance between two sites is another cost to consider for sink mobility problems because mobile sinks spend energy (for example, fuel) to move from one site to another. The distance traveled effects also the reconfiguration latency. Figure 5.3 shows

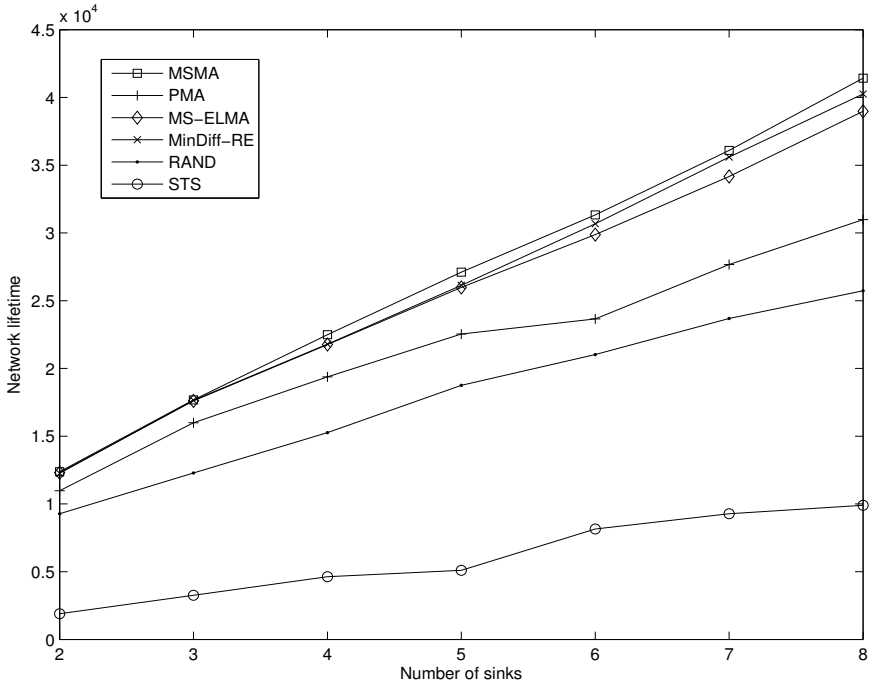

Figure 5.1: Network lifetime versus number of sinks.

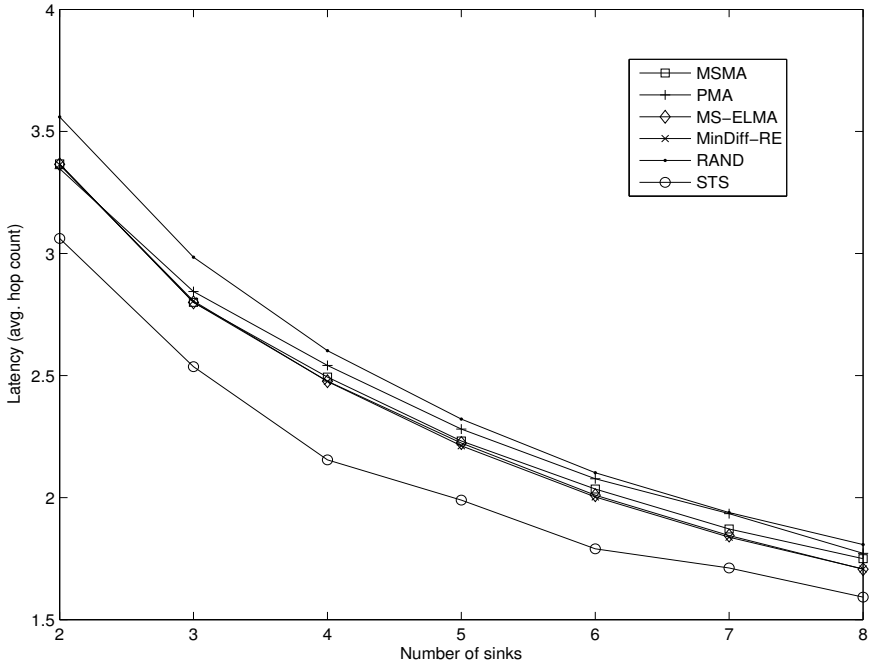

Figure 5.2: Latency versus number of sinks.

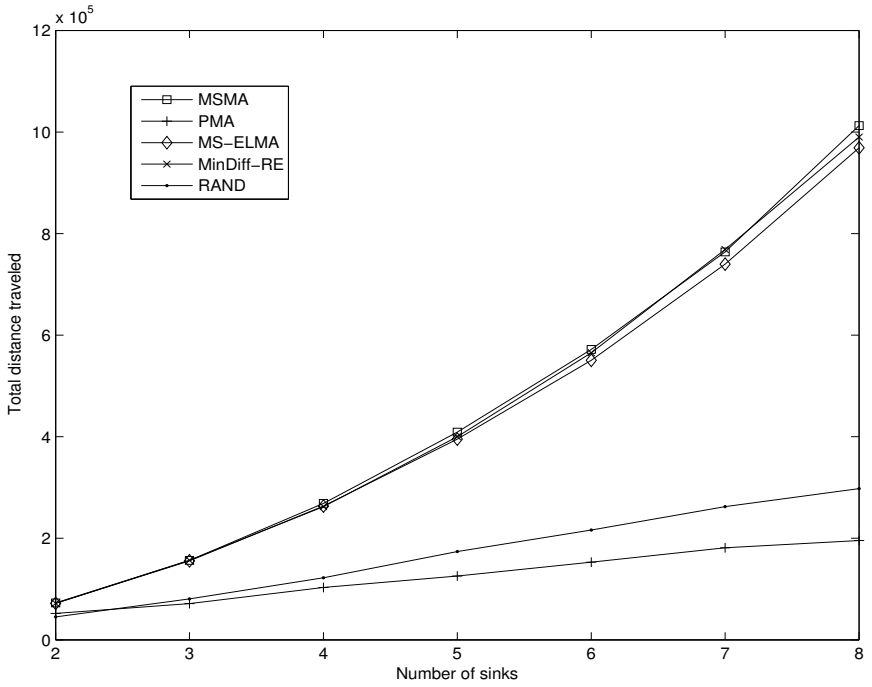

Figure 5.3: Distances traveled for various number of sinks.

the total distance covered by mobile base stations for different sink-count values. Centralized algorithms cause sinks to travel more compared to our PMA algorithm and the RAND approach. For example, when the sink-count is eight, centralized algorithms cause the sinks to cover 5.2 times more distance than the other two schemes. Our PMA algorithm achieves the least travel-distance. Sinks cover 1.5 times more distance when they use RAND approach compared to PMA algorithm. Therefore, our PMA algorithm has a longer network lifetime and better data latency while traveling less, compared to the RAND approach.

Figure 5.4 gives network lifetime performance of the schemes for different values of transmission-range. The sink-count and node-count values are fixed to 5 and 500, respectively. The performance of all schemes gets better when transmission-range increases. The performance of RAND increases by almost 50% when the transmission range is increased from 30m to 50m. This is more than the increase of any other scheme (others vary between 20% and 30%). Since RAND does not use any intelligent mechanism while moving sinks, increasing transmission range

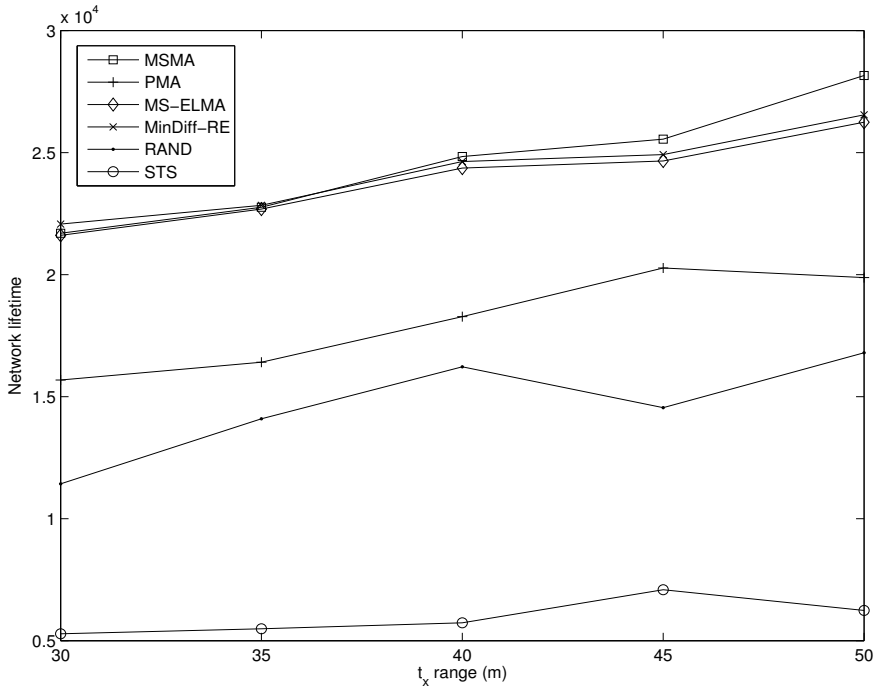

Figure 5.4: Network lifetime versus transmission ranges.

causes an increase its performance more than the performance increase of any other scheme.

The effect of the number of sink sites (site-count) on the network-lifetime performance of the algorithms is shown in Figure 5.5. The sink-count, node-count, and transmission-range values are fixed to 4, 500, and 40m, respectively. The lifetime achieved by MSMA, MinDiffRE, and MS-ELMA schemes improves 50% on average when site-count increases from 10 to 25. Increasing possible sink sites creates more options (combinations) for these algorithms to select from, which helps improving the lifetime of the network. This increase amount drops to 10% for the PMA algorithm, since it does not use any prior information (topology information for each site) while selecting the sites on-the-fly. Therefore, increasing the number of sites does not affect PMA performance as much as its centralized counterparts.

Network lifetime values for different minimum stay time (t_{min}) values are given

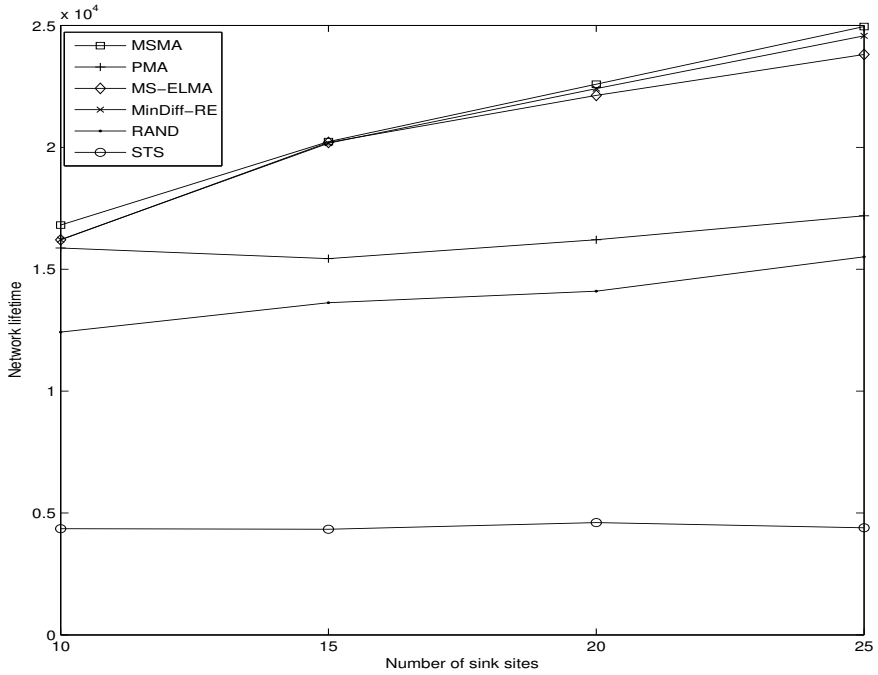

Figure 5.5: Network lifetime versus number of sink sites.

in Figure 5.6. The sink-count and site-count values are fixed to 4 and 18, respectively. The t_{min} value is changed between 50 and 250 . The performance of all algorithms is not affected by more than 7.5%. Our MSMA algorithm is affected least (2%), while RAND approach is affected most (7.5%). However, increasing t_{min} helps sinks to travel less in the area, as seen in Figure 5.7. Total travel distances decrease by an average factor of 5 while t_{min} increases from 50 to 250. That means it is possible to reduce the mobility cost by a factor of 5 when the network lifetime performance is decreased by just five percent. However, increasing t_{min} value beyond a threshold will cause algorithms' achieved lifetime values to decrease, since staying longer at a point can cause unbalanced energy values of the nodes. Therefore, ideal t_{min} value can change according to the application or needs. It should be increased if mobility cost is taken into consideration, however beyond a threshold, this will cause lifetime performance to decrease.

The effect of the number of node changed in the area is shown in Figure 5.8. Network lifetime performance is inversely affected by increasing sensor nodes in

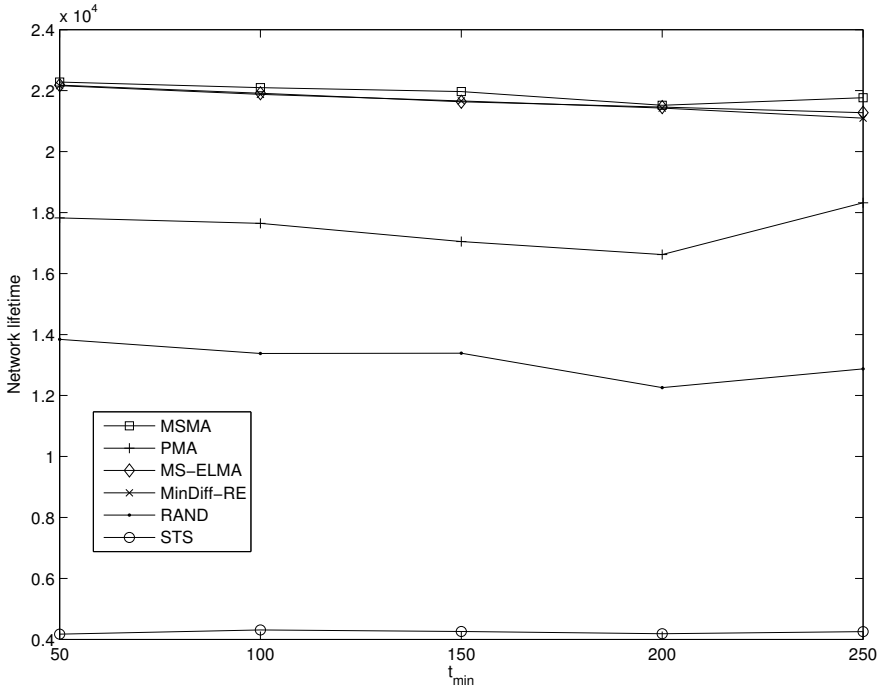

Figure 5.6: Network lifetime versus t_{min} values.

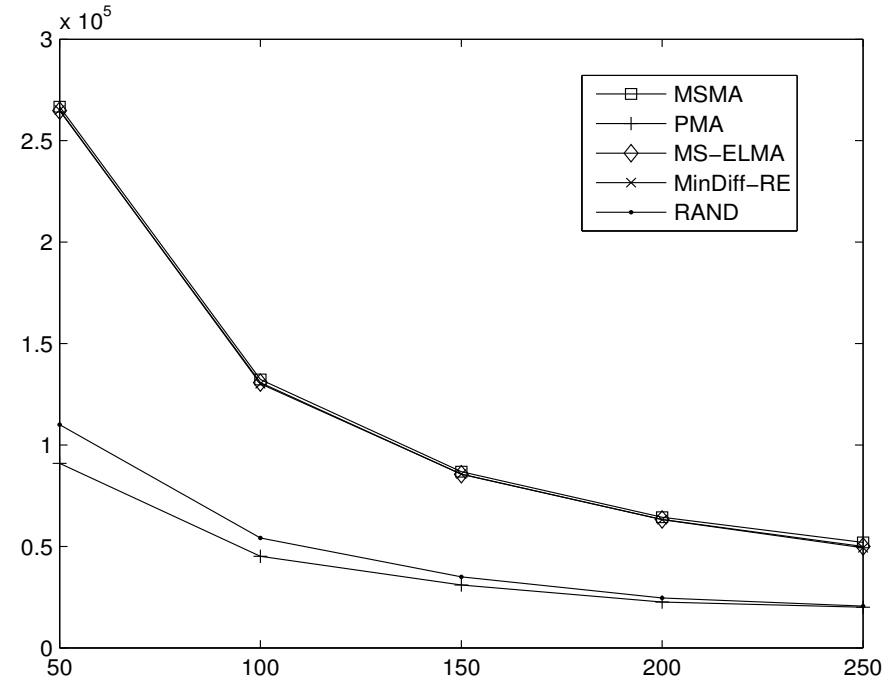

Figure 5.7: Distances traveled for various t_{min} values.

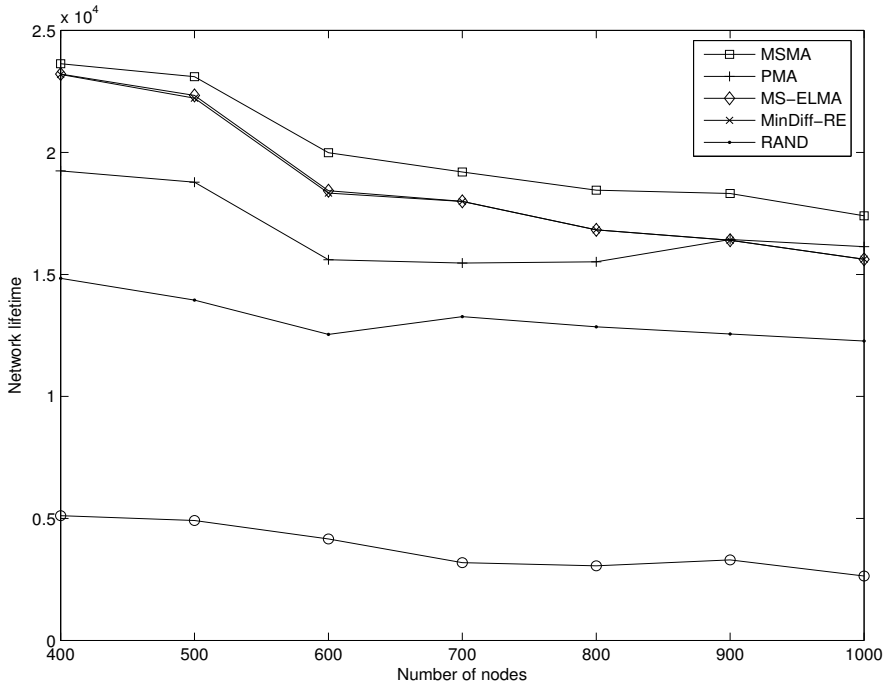

Figure 5.8: Network lifetime versus number of nodes.

the area. In this case, hot-spot nodes must relay more packets compared to when there are fewer nodes. However, the decreasing ratio in network lifetime changes according to the algorithm used. The static sink case is affected most (93%) and the PMA algorithm least (19%). The PMA's algorithm network lifetime becomes better than MinDiff-RE's and MS-ELMA's when the number of nodes increases to 1000. Since the current energy levels of the nodes are taken into consideration when the network operates, more-balanced topology trees can be constructed. This situation contributes to lifetime more than the pre-calculated topology trees used in MinDiff-RE and MS-ELMA. Therefore, it would be better to use the PMA algorithm when there are more than a few hundred nodes in the area. Sinks also travel two times less often using that method compared to centralized algorithms.

5.4 Conclusion

In this chapter, we propose two multiple-sink mobility algorithms, MSMA and PMA, to coordinate movements of multiple sinks in a wireless sensor network. The MSMA uses the energy expenditure information of all nodes for different sink-site combinations. Unlike other similar approaches, the MSMA does not use all possible combinations, but instead limits the number of combinations according to a threshold value to reduce complexity. The PMA algorithm is a distributed algorithm that uses no global network information to determine sinks' next migration points. Each sink forbids some sites from using its children's residual energy information. One group of sites is selected from the remaining set using the maximum of the minimum energy of the first-hop nodes. The other group is selected farther from the previously selected ones. We compare the performance of the proposed schemes with the MinDiff-RE, RAND, and static-sink approaches in terms of network lifetime, latency, and total distance covered by the sinks.

Under varying metrics, the experiment results show that the MSMA performs better in terms of network than any other scheme. The MSMA gives better network lifetime than the MinDiff-RE even though it uses up to four times fewer sink-site combinations. Latency (average hop count to any sink) is lowest for the static-sink case and almost the same for the other schemes. The PMA algorithm achieves less lifetime (around 20% on the average) compared to centralized algorithms (but better than random movement under all conditions), however, its performance increases (becomes better than the MinDiff-RE and MS-ELMA) when there are more than a few hundred nodes. Sinks also travel less compared to other schemes when using the PMA algorithm.

Chapter 6

Conclusions and Future Work

In this thesis, we propose different sink mobility algorithms for both single-sink and multiple-sink sensor networks. We also propose two different energy efficient routing topology construction algorithms that complement our sink mobility algorithms to further improve lifetime of wireless sensor networks.

We first present two different energy-efficient sink mobility algorithms for single-sink mobility problem. Our packet-load and energy-load based mobility algorithms, called PLMA and ELMA, respectively, construct and use load matrices which incorporate packet-count and energy expenditure values of sensor nodes for each candidate sink site. A training phase is carried out before network starts operating, in which a sink node visits each site location and constructs a routing topology to determine the parents of nodes and packet/energy load values. ELMA and PLMA use the matrix for selecting the next point to move via maximizing the minimum residual energy in the network.

We also give an integer programming model of the problem to get the optimal results using the elements of matrices as variables. The solution of this formulation gives an upper bound on network lifetime and is used as a benchmark to evaluate the performance of ELMA and PLMA algorithms. In addition to comparing the performance of our algorithms with optimum results, we also compare

them with the performance of random movement and static sink cases. Our simulation results show that our ELMA algorithm can provide up- to five times better lifetime than static sink case and two times better lifetime than random movement. ELMA's performance is just five percent below the optimal results.

Next, we give two different energy efficient routing topology construction algorithms for sensor networks where data aggregation is not used in sensor nodes. Constructing energy-efficient routing topologies improve the performance of our mobility algorithms further in terms of network lifetime. Our centralized topology construction algorithm (CTCA) extends our earlier load balance topology construction algorithm (LBTCA) and considers the load of all ancestors of alternative parents to properly balance energy consumption of nodes. Although CTCA algorithm improves the performance of LBTCA, centralized algorithms cause extra delay due to the requirement of collecting node states at the center. In our distributed topology construction algorithm (DTCA), each node selects its parent without needing to exchange messages with its neighbors and parents.

We also propose an adaptive version of our ELMA algorithm, called A-ELMA, which removes the training phase and the constraint that nodes have to select the same parent each time sink visits the same location. The algorithm learns the parameters after visiting each site randomly. It also selects sites using the information gathered about the previously visited sites. Simulation results show that A-ELMA performs better network lifetime than ELMA when using CTCA or DTCA as the routing topology construction algorithm.

Finally, we present two different sink mobility algorithms for multiple-sink mobility problem. Our Multiple Sinks Movement Algorithm (MSMA) is a centralized algorithm which limits the number of sink-site combinations to use while selecting sinks' next positions. It also selects site positions randomly, with a very small probability, to add other site combinations and take advantage of nodes' remaining energy values to construct energy-efficient trees. Our Prevent and Move Away (PMA) algorithm, on the other hand, is a totally distributed algorithm where sinks first determine and share information about disadvantageous sites to prevent these sites to be selected in the next round. Then, they select and share

sink sites for the next round. Half of the sites is selected to be the set of the sites that have energy-rich sensor nodes around them, The other half is selected to be the sites that are distance enough from the previously selected half. We compare the performance of our MSMA and PMA algorithms with performance of MinDiff-RE algorithm, random movement, and static sink case. Our simulation results show that our MSMA algorithms perform better than MinDiff-RE and other approaches even though it uses fewer site combinations. The performance of our PMA algorithm is worse than our MSMA algorithm in terms of network lifetime, but PMA requires less total movement and performs better than random movement and static sink case.

We analyzed the effect of number of nodes and transmission range on network lifetime. Lifetime is improved when number of nodes and transmission range values increase, however the impact of each parameter is different. Increasing number of nodes slightly improves the network lifetime. Nodes can find more options to send their packets which causes a balance in load distribution. However, since the number of packets also increases, this limits the amount of improvement in network lifetime. For instance, when we increase number of nodes from 400 to 1200 (300 %), improvement in lifetime becomes only around 50%. On the other hand, increasing transmission range contributes to lifetime more effectively than number of nodes. Increasing transmission range enables nodes to have more parent candidates. This causes the load to be distributed evenly around the sink while the number of nodes (packets) is constant in the environment. To give an example, increasing transmission range from 25m to 45m (80%) in our experiments yields an increase of 91% in network lifetime for our ELMA and PLMA algorithms. We also analyzed the effect of transmission range on latency. As expected, latency decreases when transmission range increases, since the average path length to the sink decreases.

We investigated the effect of number of sinks on network lifetime and latency for multiple-sink problem. Obviously, increasing the number of sinks directly improves the network lifetime. Nodes in the area are allocated to more than one sink and packet load for each sink decreases compared to single-sink case. Similarly, average hop count to the sink decreases for the same reason. We also

investigated the effect of minimum number of rounds that each sink should stay at a point, t_{min}, on network lifetime and total distance traveled by the sinks. Our algorithms' performance do not change so much by increasing t_{min}. Increasing t_{min} from 50 to 250 decreases lifetime by only around 14%. However, this causes sinks to visit almost five times less distance. Therefore, an ideal t_{min} value can be selected considering the application delay tolerance, mobility cost, and network lifetime performance.

As a future work, we consider investigating sink-mobility algorithms for energy harvesting sensor networks. So far in this thesis we assumed nodes are battery powered that are not rechargeable. However, in energy harvesting sensor networks, nodes can harvest energy from renewable resources. Sink mobility algorithms can be extended such that sinks move to the points where nodes are most likely to harvest more energy than the others for a given energy prediction model.

When sinks are highly mobile, full routing topology reconstruction can be an expensive operation in terms of both energy consumption and latency. A dynamic topology update algorithm can be designed to reduce latency and energy consumption due to excessive topology reconstructions. This may be especially useful for multi-sink sensor networks. These dynamic algorithms can connect some sensor nodes to other sinks after a sink node moves to a new location. The update messages of sink nodes, i.e., topology reconstruction broadcast messages, can be restricted to certain number of levels, such as first and second-hop neighbors, to save energy.

Bibliography

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002.

[2] "List of wireless sensor nodes." `https://en.wikipedia.org/wiki/List_of_wireless_sensor_nodes`. [Online; accessed 31-August-2015].

[3] P. Rawat, K. D. Singh, H. Chaouchi, and J. M. Bonnin, "Wireless sensor networks: a survey on recent developments and potential synergies," *The Journal of Supercomputing*, vol. 68, no. 1, pp. 1–48, 2014.

[4] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer Networks*, vol. 52, no. 12, pp. 2292–2330, 2008.

[5] J. Mcouat, *Wireless Sensor Networks: Principles, Design and Applications*. Signals and Communication Technology, Springer London, Limited, 2013.

[6] S. Roundy, D. Steingart, L. Frechette, P. Wright, and J. Rabaey, "Power sources for wireless sensor networks," in *Wireless Sensor Networks*, vol. 2920 of *Lecture Notes in Computer Science*, pp. 1–17, Springer, 2004.

[7] A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava, "Power management in energy harvesting sensor networks," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 6, no. 4, p. 32, 2007.

[8] D. Niyato, E. Hossain, M. M. Rashid, and V. K. Bhargava, "Wireless sensor networks with energy harvesting technologies: a game-theoretic approach

to optimal energy management," *IEEE Wireless Communications*, vol. 14, no. 4, pp. 90–96, 2007.

[9] S. Sudevalayam and P. Kulkarni, "Energy harvesting sensor nodes: Survey and implications," *IEEE Communications Surveys & Tutorials*, vol. 13, no. 3, pp. 443–461, 2011.

[10] M. Cardei, W. Jie, L. Mingming, and M. Pervaiz, "Maximum network lifetime in wireless sensor networks with adjustable sensing ranges," in *IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pp. 438–445, 2005.

[11] H. Yoon, B. Lee, T. Lee, and M. Chung, "Energy efficient routing with power management to increase network lifetime in sensor networks," in *Computational Science and Its Applications - ICCSA*, vol. 3046 of *Lecture Notes in Computer Science*, pp. 46–55, 2004.

[12] D. Nguyen, L. Q. V. Tran, O. Berder, and O. Sentieys, "A low-latency and energy-efficient mac protocol for cooperative wireless sensor networks," in *IEEE Global Communications Conference (GLOBECOM)*, pp. 3826–3831, 2013.

[13] M. H. S. Gilani, I. Sarrafi, and M. Abbaspour, "An adaptive CSMA/TDMA hybrid mac for energy and throughput improvement of wireless sensor networks," *Ad Hoc Networks*, vol. 11, no. 4, pp. 1297–1304, 2013.

[14] A. A. Ahmed and N. Fisal, "A real-time routing protocol with mobility support and load distribution for mobile wireless sensor networks," *International Journal of Sensor Networks*, vol. 15, no. 2, pp. 95–111, 2014.

[15] Y. Jiang, W. Shi, X. Wang, and H. Li, "A distributed routing for wireless sensor networks with mobile sink based on the greedy embedding," *Ad Hoc Networks*, vol. 20, pp. 150–162, 2014.

[16] A. Nayak and I. Stojmenovic, *Wireless Sensor and Actuator Networks: Algorithms and Protocols for Scalable Coordination and Data Communication.* New York, NY, USA: Wiley-Interscience, 2010.

[17] S. Basagni, A. Carosi, and C. Petrioli, "Controlled vs. uncontrolled mobility in wireless sensor networks: Some performance insights," in *Vehicular Technology Conference*, pp. 269–273, 2007.

[18] R. C. Shah, S. Roy, S. Jain, and W. Brunette, "Data mules: Modeling and analysis of a three-tier architecture for sparse sensor networks," *Ad Hoc Networks*, vol. 1, no. 2, pp. 215–233, 2003.

[19] D. Kim, R. Uma, B. H. Abay, W. Wu, W. Wang, and A. O. Tokuta, "Minimum latency multiple data mule trajectory planning in wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 13, no. 4, pp. 838–851, 2014.

[20] J. Luo and J.-P. Hubaux, "Joint mobility and routing for lifetime elongation in wireless sensor networks," in *IEEE International Conference on Computer Communications (INFOCOM)*, pp. 1735–1746, 2005.

[21] I. Papadimitriou and L. Georgiadis, "Energy-aware routing to maximize lifetime in wireless sensor networks with mobile sink," *Journal of Communications Software and Systems*, vol. 2, no. 2, pp. 141–151, 2006.

[22] S. Basagni, A. Carosi, E. Melachrinoudis, C. Petrioli, and Z. M. Wang, "Controlled sink mobility for prolonging wireless sensor networks lifetime," *Wireless Networks*, vol. 14, no. 6, pp. 831–858, 2008.

[23] W. Liang, J. Luo, and X. Xu, "Prolonging network lifetime via a controlled mobile sink in wireless sensor networks," in *IEEE Global Telecommunications Conference (GLOBECOM)*, pp. 1 –6, 2010.

[24] Z. Xu, W. Liang, and Y. Xu, "Network lifetime maximization in delay-tolerant sensor networks with a mobile sink," in *IEEE 8th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pp. 9 –16, 2012.

[25] K. Akkaya, M. Younis, and M. Bangad, "Sink repositioning for enhanced performance in wireless sensor networks," *Computer Networks*, vol. 49, no. 4, pp. 512–534, 2005.

[26] Z. Vincze, D. Vass, R. Vida, A. Vidács, and A. Telcs, "Adaptive sink mobility in event-driven densely deployed wireless sensor networks," *Ad Hoc & Sensor Wireless Networks*, vol. 3, no. 2-3, pp. 255–284, 2007.

[27] A. P. Azad and A. Chockalingam, "Mobile base stations placement and energy aware routing in wireless sensor networks," in *IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 264–269, 2006.

[28] M. Cardei and D.-Z. Du, "Improving wireless sensor network lifetime through power aware organization," *Wireless Networks*, vol. 11, no. 3, pp. 333–340, 2005.

[29] X. Han, L. Shu, Y. Chen, and H. Zhou, "Wx-mac: an energy efficient MAC protocol for wireless sensor networks," in *IEEE 10th International Conference on Mobile Ad-Hoc and Sensor Systems (MASS)*, pp. 423–424, 2013.

[30] B. Khan and R. Bilal, "Mobility adaptive energy efficient and low latency MAC for wireless sensor networks," *International Journal of Handheld Computing Research*, vol. 4, no. 2, pp. 40–54, 2013.

[31] M. Aissani, S. Bouznad, B. Djamaa, and I. Tsabet, "Efficient energy-aware mechanisms for real-time routing in wireless sensor networks," in *Ad-hoc, Mobile, and Wireless Networks*, pp. 304–317, 2014.

[32] K. N. Kannan and B. Paramasivan, "Development of energy-efficient routing protocol in wireless sensor networks using optimal gradient routing with on demand neighborhood information," *International Journal of Distributed Sensor Networks*, vol. 2014. Article ID 208023, 7 pages, 2014.

[33] S. Su, H. Yu, and Z. Wu, "An efficient multi-objective evolutionary algorithm for energy-aware QoS routing in wireless sensor network," *International Journal of Sensor Networks*, vol. 13, no. 4, pp. 208–218, 2013.

[34] J. Hao, G. Duan, B. Zhang, and C. Li, "An energy-efficient on-demand multicast routing protocol for wireless ad hoc and sensor networks," in *IEEE Global Communications Conference (GLOBECOM)*, pp. 4650–4655, 2013.

[35] L. Wang and Y. Xiao, "A survey of energy-efficient scheduling mechanisms in sensor networks," *Mobile Networks and Applications*, vol. 11, no. 5, pp. 723–740, 2006.

[36] S. R. Gandham, M. Dawande, R. Prakash, and S. Venkatesan, "Energy-efficient schemes for wireless sensor networks with multiple mobile base stations," in *IEEE Global Telecommunications Conference (GLOBECOM)*, pp. 377–381, 2003.

[37] J. Luo, J. Panchard, M. Piórkowski, M. Grossglauser, and J.-P. Hubaux, "Mobiroute: Routing towards a mobile sink for improving lifetime in sensor networks," in *IEEE International Conference on Distributed Computing in Sensor Systems*, pp. 480–497, 2006.

[38] A. Woo, T. Tong, and D. Culler, "Taming the underlying challenges of reliable multihop routing in sensor networks," in *1st International Conference on Embedded Networked Sensor Systems*, pp. 14–27, 2003.

[39] M. Emre Keskin, İ. Kuban Altınel, N. Aras, and C. Ersoy, "Wireless sensor network lifetime maximization by optimal sensor deployment, activity scheduling, data routing and sink mobility," *Ad Hoc Networks*, vol. 17, pp. 18–36, 2014.

[40] Y. Yun, Y. Xia, B. Behdani, and J. Smith, "Distributed algorithm for lifetime maximization in a delay-tolerant wireless sensor network with a mobile sink," *IEEE Transactions on Mobile Computing*, vol. 12, no. 10, pp. 1920–1930, 2013.

[41] L. A. Wolsey, *Integer Programming*. Wiley-Interscience, 1998.

[42] K. Kalpakis, K. Dasgupta, and P. Namjoshi, "Efficient algorithms for maximum lifetime data gathering and aggregation in wireless sensor networks," *Computer Networks*, vol. 42, no. 6, pp. 697–716, 2003.

[43] R. Velmani and B. Kaarthick, "An efficient cluster-tree based data collection scheme for large mobile wireless sensor networks," *IEEE Sensors Journal*, vol. 15, no. 4, pp. 2377–2390, 2015.

[44] W. R. Heinzelman, J. Kulik, and H. Balakrishnan, "Adaptive protocols for information dissemination in wireless sensor networks," in *International Conference on Mobile Computing and Networking*, pp. 174–185, 1999.

[45] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks," in *International Conference on Mobile Computing and Networking*, pp. 56–67, 2000.

[46] S. Lindsey and C. S. Raghavendra, "PEGASIS: Power-efficient gathering in sensor information systems," in *Aerospace Conference Proceedings*, pp. 1125–1130, 2002.

[47] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Transactions on Wireless Communications*, vol. 1, no. 4, pp. 660–670, 2002.

[48] O. Younis and S. Fahmy, "HEED: a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks," *IEEE Transactions on Mobile Computing*, vol. 3, no. 4, pp. 366–379, 2004.

[49] J. Yang, B. Bai, and H.Li, "A cluster-tree based data gathering algorithm for wireless sensor networks," in *International Conference on Automatic Control and Artificial Intelligence (ACAI)*, pp. 22–25, 2012.

[50] Z. Zhang and F. Yu, "Performance analysis of cluster-based and tree-based routing protocols for wireless sensor networks," in *International Conference on Communications and Mobile Computing (CMC)*, vol. 1, pp. 418–422, 2010.

[51] Q. Mamun, "A qualitative comparison of different logical topologies for wireless sensor networks," *Sensors*, vol. 12, no. 11, pp. 14887–14913, 2012.

[52] J. Liang, J. Wang, J. Cao, J. Chen, and M. Lu, "An efficient algorithm for constructing maximum lifetime tree for data gathering without aggregation in wireless sensor networks," in *IEEE Conference on Computer Communications (INFOCOM)*, pp. 1–5, 2010.

[53] D. Luo, X. Zhu, X. Wu, and G. Chen, "Maximizing lifetime for the shortest path aggregation tree in wireless sensor networks," in *IEEE Conference on Computer Communications (INFOCOM)*, pp. 1566–1574, 2011.

[54] H.-C. Lin, F.-J. Li, and K.-Y. Wang, "Constructing maximum-lifetime data gathering trees in sensor networks with data aggregation," in *IEEE International Conference on Communications (ICC)*, pp. 1–6, 2010.

[55] H. Zhang and H. Shen, "Balancing energy consumption to maximize network lifetime in data-gathering sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 20, no. 10, pp. 1526–1539, 2009.

[56] W. Liang and Y. Liu, "Online data gathering for maximizing network lifetime in sensor networks," *IEEE Transactions on Mobile Computing*, vol. 6, no. 1, pp. 2–11, 2007.

[57] H. O. Tan and I. Korpeoglu, "Power efficient data gathering and aggregation in wireless sensor networks," *ACM Sigmod Record*, vol. 32, no. 4, pp. 66–71, 2003.

[58] T.-W. Kuo and M.-J. Tsai, "On the construction of data aggregation tree with minimum energy cost in wireless sensor networks: NP-completeness and approximation algorithms," in *IEEE Conference on Computer Communications (INFOCOM)*, pp. 2591–2595, 2012.

[59] H. Lee, A. Keshavarzian, and H. Aghajan, "Near-lifetime-optimal data collection in wireless sensor networks via spatio-temporal load balancing," *ACM Transactions on Sensor Networks (TOSN)*, vol. 6, no. 3, p. 26, 2010.

[60] C. Hua and T.-S. P. Yum, "Optimal routing and data aggregation for maximizing lifetime of wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 16, no. 4, pp. 892–903, 2008.

[61] S. Xiong, J. Li, and L. Yu, "Maximize the lifetime of a data-gathering wireless sensor network," in *IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, pp. 1–9, IEEE, 2009.

[62] D. Li, Q. Zhu, and W. Chen, "Efficient algorithm for maximum lifetime many-to-one data aggregation in wireless sensor networks," *International Journal of Sensor Networks*, vol. 9, no. 2, pp. 61–68, 2011.

[63] Y. Wu, S. Fahmy, and N. B. Shroff, "On the construction of a maximum-lifetime data gathering tree in sensor networks: NP-completeness and approximation algorithm," in *IEEE Conference on Computer Communications (INFOCOM)*, 2008.

[64] X. Zhu, X. Wu, and G. Chen, "An exact algorithm for maximum lifetime data gathering tree without aggregation in wireless sensor networks," *Wireless Networks*, vol. 21, no. 1, pp. 281–295, 2015.

[65] J. Yuan, H. Zhou, and H. Chen, "Constructing maximum-lifetime data gathering tree without data aggregation for sensor networks," in *Computer and Information Science 2011*, vol. 364 of *Studies in Computational Intelligence*, pp. 47–57, 2011.

[66] L. Carr-Motyčková and D. Dryml, "Distributed energy efficient data gathering without aggregation via spanning tree optimization," in *Ad-hoc, Mobile, and Wireless Network*, vol. 7960 of *Lecture Notes in Computer Science*, pp. 87–98, 2013.

[67] S. Gandham, M. Dawande, R. Prakash, and S. Venkatesan, "Energy efficient schemes for wireless sensor networks with multiple mobile base stations," in *Global Telecommunications Conference*, pp. 377 – 381, 2003.

[68] S. Basagni, A. Carosi, C. Petrioli, and C. Phillips, "Coordinated and controlled mobility of multiple sinks for maximizing the lifetime of wireless sensor networks," *Wireless Networks*, vol. 17, pp. 759–778, 2011.

[69] W. Liang and J. Luo, "Network lifetime maximization in sensor networks with multiple mobile sinks," in *IEEE 36th Conference on Local Computer Networks (LCN)*, pp. 350–357, 2011.

[70] L. Friedmann and L. Boukhatem, "Efficient multi-sink relocation in wireless sensor network," in *Third International Conference on Networking and Services (ICNS)*, pp. 90–97, 2007.

[71] R. Silva, J. S. Silva, and F. Boavida, "Mobility in wireless sensor networks–survey and proposal," *Computer Communications*, vol. 52, pp. 1–20, 2014.

[72] M. Koc and I. Korpeoglu, "Controlled sink mobility algorithms for wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 2014. Article ID 167508, 12 pages, 2014.

[73] "The Sensor Network Museum." `http://www.snm.ethz.ch/`. [Online; accessed 31-August-2015].

[74] "Advanticsys SG 1000." `http://www.advanticsys.com/shop/assg1000-p-28.html?language=en/`. [Online; accessed 31-August-2015].

[75] H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack Problems*. Springer, Berlin, 2004.

[76] M. J. Magazine and M.-S. Chern, "A note on approximation schemes for multidimensional knapsack problems," *Mathematics of Operations Research*, vol. 9, no. 2, pp. 244–247, 1984.

[77] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *33rd International Conference on System Sciences*, pp. 1–10, 2000.

[78] "IBM CPLEX Optimizer." `http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/`. [Online; accessed 31-August-2015].

[79] R. Hoes, T. Basten, W.-L. Yeow, C.-K. Tham, M. Geilen, and H. Corporaal, "QoS management for wireless sensor networks with a mobile sink," in *Wireless Sensor Networks*, vol. 5432 of *Lecture Notes in Computer Science*, pp. 53–68, 2009.

[80] G. Wang, T. Wang, W. Jia, M. Guo, and J. Li, "Adaptive location updates for mobile sinks in wireless sensor networks," *The Journal of Supercomputing*, vol. 47, no. 2, pp. 127–145, 2009.

[81] K. Lee, "A data gathering scheme using mobile sink dynamic tree in wireless sensor networks," in *IT Convergence and Services*, vol. 107 of *Lecture Notes in Electrical Engineering*, pp. 99–107, 2011.

[82] M. Koc and I. Korpeoglu, "Traffic- and energy-load–based sink mobility algorithms for wireless sensor networks," *International Journal of Sensor Networks*. In Press.

[83] R. E. Burkard, M. Dell'Amico, and S. Martello, *Assignment Problems, Revised Print*. SIAM, 2012.