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

Metin Koç

Department of Computer Engineering,
Bilkent University, 06800, Ankara, Turkey
Email: mkoc@cs.bilkent.edu.tr
*Corresponding author

Ibrahim Korpeoglu

Department of Computer Engineering,
Bilkent University, 06800, Ankara, Turkey
Email: korpe@cs.bilkent.edu.tr

Abstract: Moving the sink node is an effective solution for improving the lifetime of wireless sensor networks (WSN). Different methods in the literature schedule sink movements and determine sink stay points. This paper provides another insight to the sink mobility problem in WSNs by incorporating node-load parameters into a matrix and using this matrix to determine which sink site to visit in each round. We first present a packet- (traffic) load–based sink movement algorithm that relies on the packet distribution of nodes in each sink site for a given topology construction algorithm. We extend this algorithm by considering the distances the packets are transmitted, and in this manner obtain an energy-load–based algorithm. We also provide an integer programming (IP) model to compute optimal results. Our extensive simulations show that our energy- and packet-based algorithms significantly improve network lifetime compared to keeping the sink static or moving it randomly. Our energy-based algorithm can increase network lifetime by a factor of 2 compared to random movement and by a factor of 5 compared to keeping the sink static. It remains only around 5% behind the optimal solution.

Keywords: Sink Mobility; Network Lifetime Improvement; Wireless Sensor Networks; Energy efficiency

Reference to this paper should be made as follows: Koç, M. and Korpeoglu, I. (2015) 'Traffic- and Energy-Load–Based Sink Mobility Algorithms for Wireless Sensor Networks', *Int. J. Sensor Networks*, Vol. x, No. x, pp.xxx–xxx.

Biographical notes: Metin Koç received B.S. degree from Yeditepe University, Istanbul, Turkey, M.S. degree from Bilkent University, Ankara, Turkey, all in computer engineering, in 2004, and 2008, respectively. He is currently a Ph.D. candidate in the Computer Engineering Department of Bilkent University. His current research interests include wireless ad hoc and sensor networks, distributed systems, and computer networks.

Ibrahim Korpeoglu received his Ph.D. and M.S. degrees from University of Maryland at College Park, both in Computer Science. He is currently an associate professor in the Computer Engineering Department of Bilkent University, Ankara, Turkey. Prior to joining Bilkent University, he worked in Ericsson, IBM T.J. Watson Research Center, Bell Labs, and Telcordia Technologies, in USA. He has served on the program committees of several conferences and published numerous papers in the area of computer networking. His research interests include computer networks, wireless ad hoc and sensor networks, wireless mesh networks, distributed systems, and P2P networks.

1 Introduction

A wireless sensor network (WSN) is composed of numerous tiny sensor nodes and a more powerful special node called a sink node (Akyildiz et al., 2002; Perkins, 2000). Data communication is usually many-to-one, such that all data packets generated by sensor nodes are destined for the single sink node. Because of limited energy resources, the most important issue in WSNs is energy efficiency to prolong the network lifetime; and this is also the main concern in developing algorithms and protocols for a network. Different energy–efficient approaches in various network layers are proposed in the literature for improving network lifetime, such as power control mechanisms (Cardei et al., 2005; Zongkai et al., 2004) (physical layer), energy–efficient MAC layer protocols (Nguyen et al., 2013; Gilani et al., 2013) (data link layer), and routing protocols (Ahmed and Fisal, 2014; Jiang et al., 2014) (network layer).

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 (Nayak and Stojmenovic, 2010). Making the sink mobile can help solving this problem via distributing the hot-spot load among all sensor nodes, in this way improving the network lifetime.

So far sink mobility problem is generally handled as an optimization problem and mathematical programming based solutions are given to solve it. Although it is important to see upper bounds on the network lifetime, most of these solutions do not scale and also can not 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 paper, 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 collects these parameters into a matrix structure to schedule sink movements in each round.

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., our 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 the IP formulation and solution of the problem to use as a benchmark for comparing the performance of our algorithms.

We performed extensive simulation experiments to compare our algorithms with random sink movement, with a min-max approach (the sink collects minimum remaining energy values from each site and moves to the one with the maximum energy) and with an optimally placed static sink case. The results show that the 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. The ELMA is just below five percent of the optimal value, however it has better latency values compared to the optimal and lower running time values. Lastly, we adapt the solution to the multiple-sinks case, where we have multiple mobile sinks that can move to utilize energy in a better manner. In this case, the our matrix rows become possible combinations of sink sites. Our simulation results for this case show that the ELMA algorithm performs better than the random movement method up to a factor of 2.1. This ratio increases to 2.6 when the nodes are not uniformly deployed.

The rest of the paper is organized as follows: We discuss the related work in Section 2. In Section 3, we present the network model and our algorithms. In Section 4, we give and discuss our performance evaluation results. Finally, we conclude the paper in Section 5.

2 Related Work

The most important characteristics 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 the literature to minimize the network's total energy consumption and thus improve network lifetime: adjusting the transmit power depending on the distance to the receiver when sending messages (Cardei and Du, 2005), developing energy-efficient MAC or routing protocols (Han et al., 2013; Khan and Bilal, 2013; Aissani et al., 2014; Kannan and Paramasivan, 2014; Su et al., 2013; Hao et al., 2014), minimizing the number of messages traveling in the network, and putting some sensor nodes into sleep mode and using only a necessary set for sensing and communication Wang and Xiao (2006).

Moving the sink node in the deployment region is another approach to prolonging network lifetime because this eliminates the hot-spot problem around the sink (Vincze et al., 2007; Akkaya et al., 2005).

Gandham et al. (2003) 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 (maximum three), 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 value does not improve lifetime duration (since at that time there is a sufficient number of base stations in the network such that each sensor node can transmit messages via single-hop communication).

Mobility and routing are considered together in Luo and Hubaux (2005). It is assumed that sensors are densely deployed (with a Poisson distribution) within a circle. The authors define the network lifetime as the time span until the first *loss of coverage*. They define the problem with a linear programming (LP) formulation (minimizing the load on each sensor node N) and solve it, first finding the optimal mobility strategy by fixing the routing strategy as shortest path, then using the output strategy to find a final routing strategy 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 using this structure. However, because they do not compare their results to any other mobility approach we cannot comment on the performance of their proposed scheme. The work's main drawback is the assumption that the network region is circular; there is no explanation of how the solution may be transferred to other region types.

A work more similar to ours is presented in Papadimitriou and Georgiadis (2006). In this paper, N sensor nodes and a sink node s are randomly deployed to an area of interest. There is a constant information generation rate at every sensor node and a set of locations where the base station can move and stay. 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 for each packet from a 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 compares mobile and static sink approaches with different routing schemes. The simulations use two scenarios, including just four (centers of four sub-squares) and five different (corners and center) sink sites, respectively. The authors perform and compare the experiments via adding a routing parameter, which prevents us from observing the performance of their proposed mobility model.

In Luo and Hubaux (2006), the authors give a routing algorithm, *mobiroute*, to route data towards a mobile sink for improving network lifetime. Mobiroute extends Woo et al. (2003)'s MintRoute by adding functions to perform the operations; notifying a node for link breakage, informing the whole network about topological changes, minimizing the packet loss during mobility. They propose a two phase adaptive algorithm to control sink mobility. In *initialization* (phase 1), mobile sink builds a power consumption profile for each anchor point and drops some of them if their weights are low. In *operation* (phase 2), 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 Basagni et al. (2008) 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 same problem. The network model is similar to the one given in Papadimitriou and Georgiadis (2006), 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. The d_{max} parameter represents an upper bound for the distance the sink can travel between the current and the next site. The other is t_{min}, where it defines mandatory time the sink is required to stay at any site. The authors evaluate the performance of MILP, GMRE, random movement (RM), and static sink (STS) approaches. 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 Keskin et al. (2014). The authors give a mixed integer linear programming model, which addresses the aforementioned parameters together to maximize network lifetime. However, the authors present two heuristics since current state-of-the-art MILP solvers fail to solve problems with realistic size due to time limitations. 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. Both heuristics give relatively longer lifetime compared to MILP solver especially for larger number of sensor locations.

Although centralized algorithms are proposed in previous works, Yun et al. (2013) present 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 main algorithm and its 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, Gandham et al. (2003) uses maximum 30 nodes while Papadimitriou and Georgiadis (2006) uses 100 nodes. Since IP or MILP is NP complete (Wolsey, 1998), 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), and 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 also more effective than the optimal solution. Therefore in this paper, we provide heuristic algorithms which can be used at network operation time while maintaining a current situation matrix and giving decisions based on that instead of performing long and static pre-calculations.

3 Proposed Solution

3.1 System Model

We consider a wireless sensor network that has N static sensor nodes and a mobile base station (sink). Sensor nodes are deployed to the region of interest in a random manner. There is a *training* phase, as in Basagni et al. (2011), such that mobile sink visits all sink sites once before network starts its operation. After the mobile sink moves to a location, it stays there for a while and constructs a routing tree by initiating the flooding of a control packet which includes its id and hop count (both are zero for the sink) in the network. Each node receives the packet sets its parent (if multiple such packets are received, node with smaller hop count - smaller id in case of a tie - is selected as a parent) and re-broadcasts the packet after incrementing the hop count and 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, such that parents can calculate how many packets it will relay when sink at 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 it in the energy model to calculate energy expenditure for 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 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 negligible as in (Basagni et al., 2008; Papadimitriou and Georgiadis, 2006; Yun et al., 2013)). 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 above, when a sink moves to a point it constructs a tree-based routing structure (routing topology) rooted at the sink node for each sink site during *training* phase as explained in the previous section. In this phase, a packet traffic load matrix T can be 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).

$$T = \begin{array}{c} \\ p_1 \\ p_2 \\ \vdots \\ p_m \end{array} \begin{array}{c} \begin{array}{cccc} o_1 & o_2 & \dots & o_n \end{array} \\ \begin{pmatrix} t_{11} & t_{12} & \dots & t_{1n} \\ t_{21} & t_{22} & \dots & t_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ t_{m1} & t_{m2} & \dots & t_{mn} \end{pmatrix} \end{array}$$

The packet load matrix is constructed before the network begins operation. But before the matrix is constructed we need to know the possible sink migration positions, which can be determined by various algorithms (Koc and Korpeoglu, 2014). 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 the 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 (parent). Hence, our second algorithm is based on nodes' energy expenditures for a given sink position.

Next we provide the details of our algorithm.

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.

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

```
1:  procedure PLMA($c, t_x$)  ▷ c: node coordinates, $t_x$: transmission range
2:      $p \leftarrow getMigrationPoints(c, t_x)$
3:      for $i \leftarrow 1, size(p)$ do
4:          $tt \leftarrow constructTopology(c, t_x, p_i)$
5:          for $j \leftarrow 1, n$ do                ▷ for each node
6:              $t_{ij} \leftarrow k_i$  ▷ $k_i$: # of packets to send for $o_j$
7:          end for
8:      end for
9:      for $i \leftarrow 1, size(p)$ do
10:         $mp(i) \leftarrow max(t_i)$ ▷ max element for each row
11:     end for
12:     $r \leftarrow min(mp)$ ▷ index of minimum of maximums
13:     $cpl \leftarrow t_r$              ▷ initialize current packet list
14:     while $e > 0$ do                 ▷ all nodes are alive
15:         for each round do
16:             for $i \leftarrow 1, size(p)$ do
17:                 $u_i \leftarrow t_i + cpl$  ▷ current packet matrix
18:             end for
19:             for $i \leftarrow 1, size(p)$ do
20:                 $mp(i) \leftarrow max(u_i)$  ▷ max for each row
21:             end for
22:             $r \leftarrow min(mp)$         ▷ index of min of max
23:             $cpl \leftarrow u_r$      ▷ initialize current packet list
24:         end for
25:     end while
26: end procedure
```

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 values in each row, selects the row (i.e., the next migration point) with the minimum value, and takes this row as the *current packet list*. We call this the min-max search. After the sink migrates to that point, it would operate there 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 (matrix U). Then it again runs the min-max search over U to determine the new packet list value. This iteration continues until a node depletes its energy.

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.

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 (overhead) becomes $p\,log(N)$ bits. For 500 nodes and 30 sink sites, it requires around 34 bytes to keep this array. It is quite acceptable, since most of the nodes have more than 32 KB program and data memories (Sensor Network Museum, http://www.snm.ethz.ch). Sink maintains a matrix of $p\,N$ which requires $p\,N\,log(N)$. For 500 nodes and 30 sink sites, it requires around 16 KB memory to store this matrix. Sink nodes have more resources than nodes, for instance, a sink instance (gateway model) has more than 1 GB RAM capacity (Advanticsys SG 1000, http://www.advanticsys.com/).

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 (parent) in the routing tree for that sink position i. Then matrix U is updated to reflect the nodes' remaining energy values after each 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 (e.g., if an event occurs some nodes might send more packets, or some packets might be lost due to anomalies in the network). This model is more accurate than the packet-based approach because it takes distance into consideration (via the energy 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).

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

Algorithm 2 Energy-Load–Based Movement Algorithm (ELMA)

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

value in microjoule. For 500 nodes, 30 sink sites, and 10K microjoule it requires around 25 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 (Kellerer et al., 2004). 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:

$$\max \quad \sum_{i=1}^{m} x_i \tag{1}$$

$$\text{s.t.} \quad \sum_{i=1}^{m} t_{ij} x_i \le e, \quad j = 1, \dots, n$$

$$x_i \ge 0, \quad x_i \text{ integer, for all} \quad i = 1, \dots, m$$

It is shown that solving the unbounded multi-dimensional knapsack problem is *NP-complete* (Magazine and Chern, 1984). 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 Sinks 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) rooted trees.

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

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

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 below than a few thousands (i.e., 30 sink sites, three sinks; 17 sink sites, 12 sinks, etc.), which is called a *binomial threshold* (bt). Bt is a design parameter and should be determined by considering the matrix dimension that sensor nodes can store with current technology. For a higher number of sink sites, we can choose a subset of migration points (ct, which is the maximum integer where $\binom{ct}{s} \le bt$) 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 MS-PLMA and MS-ELMA, 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 $\binom{p}{s} > bt$ then            ▷ bt: binomial threshold
4:         $p' = kmeans(p, ct)$        ▷ ct max int, $\binom{ct}{s} \leq bt$
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 $\binom{p}{s} > bt$ then            ▷ bt: binomial threshold
4:         $p' = kmeans(p, ct)$        ▷ ct max int, $\binom{ct}{s} \leq bt$
5:     end if
6:     $ELMA(c, t_x, \binom{p'}{s})$
7: end procedure
```

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.

4.1 Simulation Parameters

The sensor networks generated in our simulations have N static sensor nodes and one mobile base station (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 300x300 m^2 or 400x400 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 multiple sinks experiments.

The energy model and the radio characteristics used in the simulations originate from Heinzelman et al. (2000). 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 $\frac{E_b}{E_n}$ (Heinzelman et al., 2000). Each sensor node has an energy of 10 J initially. Energy information can be represented with fourteen bits and can be carried to the sink as piggybacked information to the data packets. The parameters of our simulations and their typical values are summarized in Table 1.

All our simulations are done in the MATLAB environment. Our IP model is solved with CPLEX Optimizer supplied through IBM Academic Initiative program (IBM CPLEX Optimizer, http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/). The CPLEX for the MATLAB feature of this software provides an API, which helps solve the problem in the same MATLAB simulation environment.

4.2 Simulation Results

Network lifetime values of the various schemes for numbers of nodes between 400 and 800 are given in

Parameter	**Value**
Area	300x300 m^2
Number of Sensor Nodes	400,500,600,700,800
Node Deployment	random and uniform
Transmission Range	25 m
Data Routing	Tree based (Koc and Korpeoglu, 2014)
Sink Site Determination	Neighborhood SSDA (Koc and Korpeoglu, 2014)
Nodes' Initial Energy	10 J
Radio Characteristics	First-order radio model (Heinzelman et al., 2000)

Table 1 Simulation Parameters

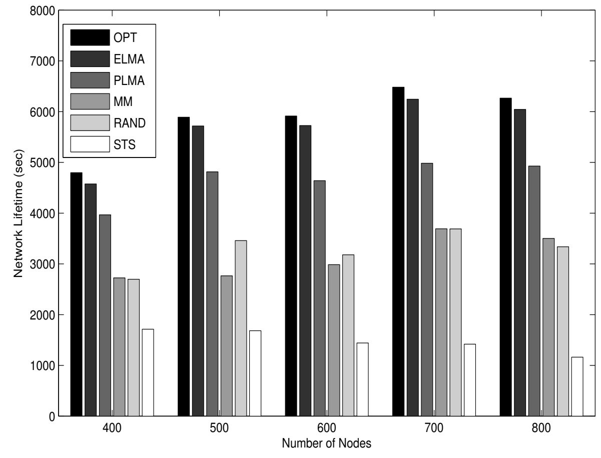

Figure 1: Network lifetime for various numbers of nodes.

Figure 1. 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 (nodes not uniformly distributed). Figure 2 shows the experiment results for various number of nodes, which although to the previous results, exhibit two differences: 1) Random movement performs worse in skew deployment compared to uniform deployment, because the random points may have been poorly selected (may reside in a sparse area, for example). Our ELMA performs at most 80% better in uniform deployment; however, 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.

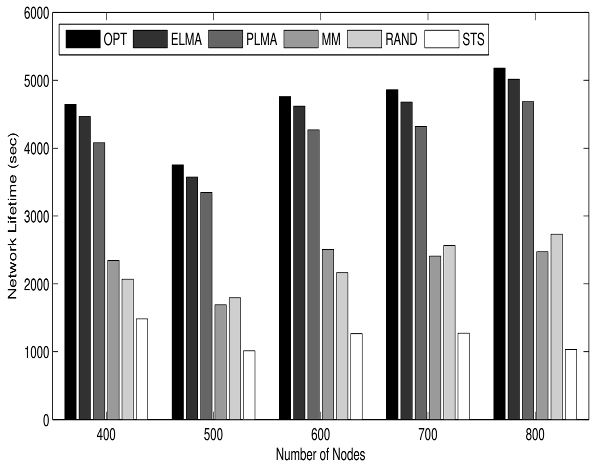

Figure 2: Network lifetime for various numbers of nodes under skew deployment.

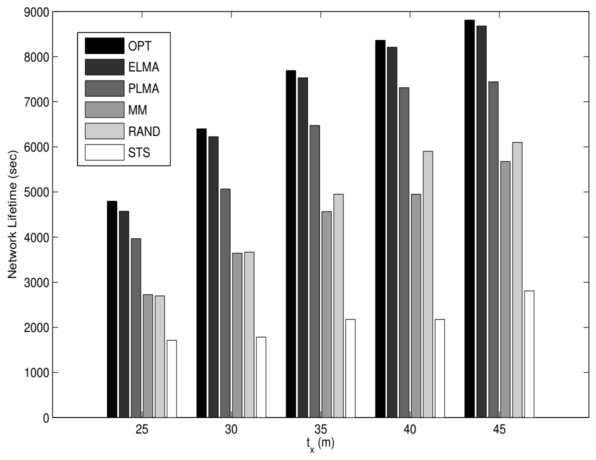

Figure 3: Network Lifetime values for various transmission ranges.

In Figure 3, 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 4. Latency values of the ELMA and PLMA and the OPT case (in terms of lifetime) are very close to each other, while the random scheme has slightly better latency values (around five percent lower). The static sink 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.

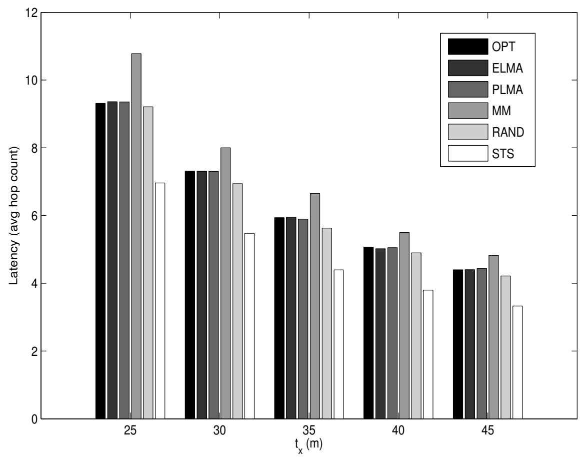

Figure 4: Latency values for various transmission ranges.

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

Table 2 Average running time values (in seconds) of two approaches

Average running time values for the OPT and ELMA are given in Table 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 a sink before the network starts operating, but the result would take a very long time to determine, and this method would give less than a five percent lifetime improvement compared to the ELMA.

The optimal algorithm 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. The ELMA, then, runs faster, can adapt to cases when unexpected packet transmissions occur, and sacrifices less than five percent of the network lifetime in doing so.

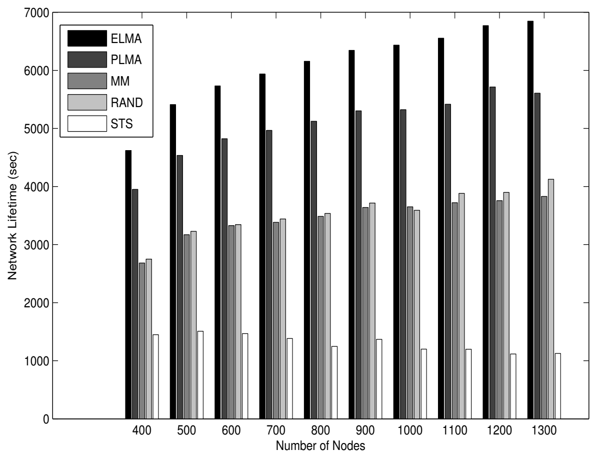

Figure 5: Network lifetime values for numbers of nodes.

Network lifetime values for higher numbers of nodes (without optimal values) are given in Figure 5. Sink mobility efficiency increases directly proportional to the number of nodes. The energy-based approach has 3.2 times more network lifetime when the nodes number 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 6. The number of sink sites is fixed to 15. Since all possible sink placements are 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 static places for sinks are determined using the output. 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 sinks by 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 the sensor nodes is more balanced than in the single-sink case. This degrades the effect of mobility because its main aim is to decrease unbalanced load distribution among sensor nodes.

Latency values for different numbers of sinks are given in Figure 7. 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% on average, while the single sink 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 8. The number of sinks is fixed to three in the experiment. Networks have better lifetime values when the number of sink sites increases

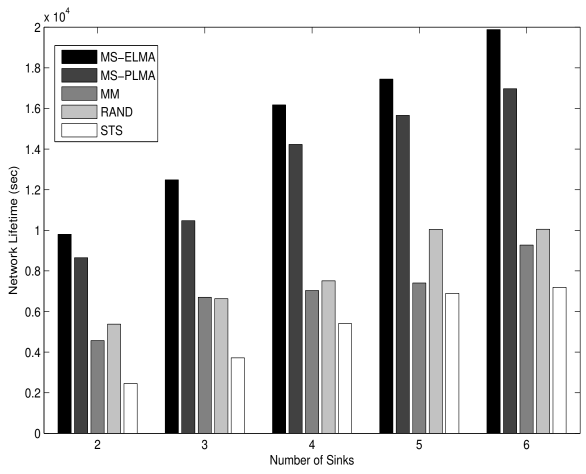

Figure 6: Network lifetime values for various numbers of sinks.

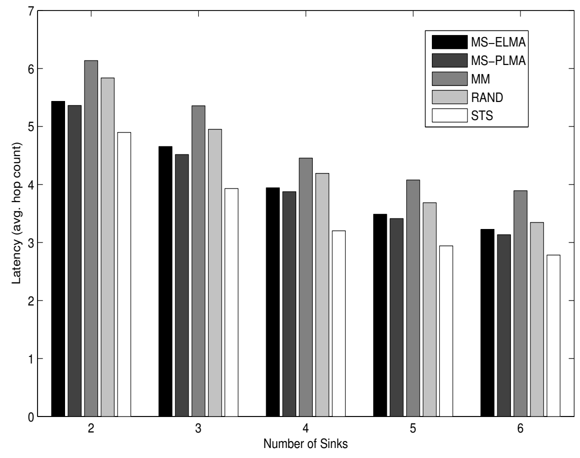

Figure 7: Latency values for various numbers of sinks.

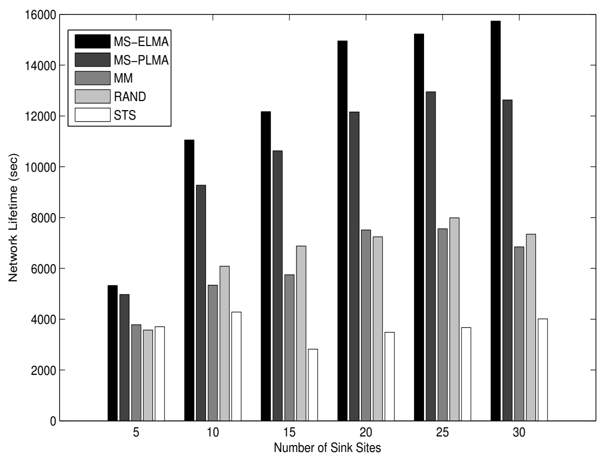

Figure 8: Network lifetime values for various numbers of sink sites.

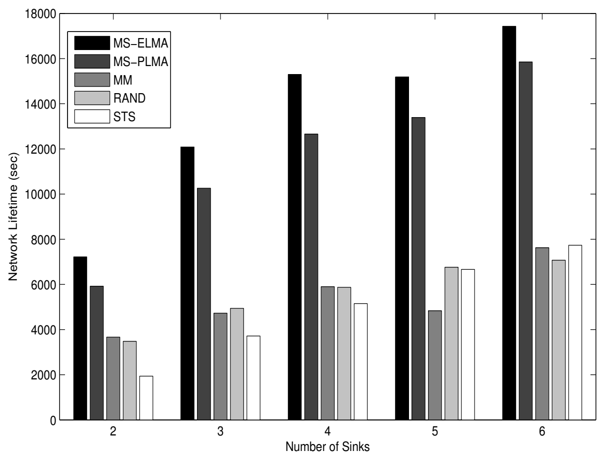

Figure 9: Network lifetime values for various numbers of sinks under skew deployment.

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 9. 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.

Distance-traveled values for different numbers of sinks under random deployment are given in Figure 10. Sinks traveled a minimum amount of distance using the min-max algorithm (24% less compared to others on the

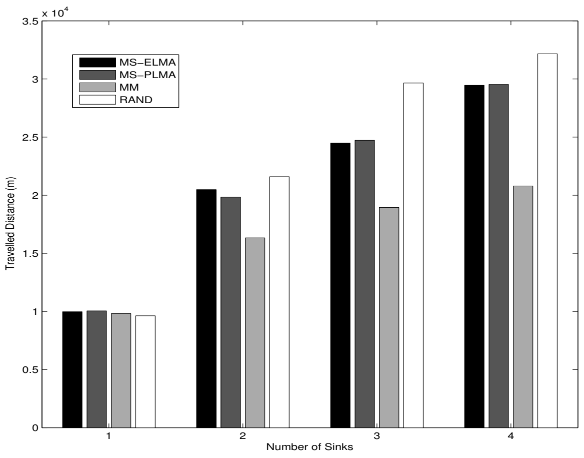

Figure 10: Distance traveled for various numbers of sinks.

average). The MS-ELMA and MS-PLMA have similar values and result in less distance traveled (seven percent) compared to the random movement algorithm.

5 Conclusion

In this paper 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 to 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.

Acknowledgement

This work is supported in part by The Scientific and Technological Research Council of Turkey (TUBITAK) with project 113E274.

References

Akyildiz, I.F. , Su, W., Sankarasubramaniam, Y. and Cayirci, E. (2002) 'Wireless Sensor Networks: A Survey', *Computer Networks (Elsevier)*, Vol. 38, No. 4, pp.393–422

Perkins, C. 2000 'Ad Hoc Networks', *Addison-Wesley*

Cardei, M., Jie, W., Mingming, L. and Pervaiz, M.O. (2005) 'Energy Efficient Routing with Power Management to Increase Network Lifetime in Sensor Networks', *IEEE Intl. Conf. on Wireless and Mobile Computing, Networking and Communications (WiMob).*

Zongkai, Y., Dasheng, Z., Wenqing, C. and Jianhua, H. (2004) 'Maximum network lifetime in wireless sensor networks with adjustable sensing ranges', *Computational Science and Its Applications - ICCSA.*

Nguyen, D., Le-Quang-Vinh, T., Berder, O. and Sentieys, O. (2013) 'A Low-Latency and Energy-Efficient MAC Protocol for Cooperative Wireless Sensor Networks', *Global Communications Conference (Globecom).*

Gilani, M.H.S. , Sarrafi, I. and Abbaspour, M. (2013) '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.

Ahmed, A.A. and Fisal, N. (2014) 'A realtime 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.

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

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

Cardei, M. and Du, D. Z. (2005) 'Improving wireless sensor network lifetime through power aware organization', *Wireless Networks*, Vol. 11, No. 3, pp.333–340.

Wang, L. and Xiao, Y. (2006) 'A survey of energy-efficient scheduling mechanisms in sensor networks', *Mobile Networks and Applications*, Vol. 11, No. 5, pp.723–740.

Vincze, Z. , Vass, D., Vida, R., Vidács, A. and Telcs, A. (2007) 'Adaptive Sink Mobility in Event-Driven Densely Deployed Wireless Sensor Networks', *Ad Hoc & Sensor Wireless Networks*, Vol. 3, pp.255–284.

Han, X. , Shu, L., Yuanfang, C. and Zhou, H. (2013) 'WX-MAC: An Energy Efficient MAC Protocol for Wireless Sensor Networks', *10th International Conference on Mobile Ad-Hoc and Sensor Systems (MASS).*

Khan, B.M. and Bilal, R. (2013) '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.

Aissani, M., Bouznad, S., Djamaa, B. and Tsabet, I. (2014) 'Efficient Energy-Aware Mechanisms for Real-Time Routing in Wireless Sensor Networks', *Ad-hoc, Mobile, and Wireless Networks - 13th International Conference, ADHOC-NOW.*

Kannan, K.N. and Paramasivan, B. (2014) '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.

Su, S., Yu, H. and Wu, Z. (2013) 'An efficient multiobjective evolutionary algorithm for energyaware QoS routing in wireless sensor network', *International Journal of Sensor Networks*, Vol. 13, No. 4, pp.208–218.

Hao, J., Duan, G., Zhang, B. and Li, C. (2013) 'An energy-efficient on-demand multicast routing protocol for wireless ad hoc and sensor networks', *Global Communications Conference (GLOBECOM).*

Akkaya, K. , Younis, M. and Bangad, M. (2005) 'Sink repositioning for enhanced performance in wireless sensor networks', *Computer Networks*, Vol. 49, No. 4, pp.512–534.

Gandham, S. R., Dawande, M., Prakash, R. and Venkatesan, S. (2003) 'Energy-Efficient Schemes for Wireless Sensor Networks with Multiple Mobile Base Stations', *IEEE Global Telecommunications Conference*, pp.377–381.

Luo, J. and Hubaux, J. P. (2005) 'Joint Mobility and Routing for Lifetime Elongation in Wireless Sensor Networks', *IEEE INFOCOM*, pp.1735–1746.

Papadimitriou, I. and Georgiadis, L. (2006) '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.

Luo, J. and Hubaux, J. P. (2006) 'Mobiroute: Routing towards a mobile sink for improving lifetime in sensor networks', *IEEE International Conference on Distributed Computing in Sensor Systems*, pp.480–497.

Woo, A., Tong, T. and Culler, D. (2003) 'Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks', *1st International Conference on Embedded Networked Sensor Systems*, pp.14–27.

Basagni, S., Carosi, A., Melachrinoudis, E., Petrioli, C. and Wang, Z. M. (2008) 'Controlled Sink Mobility for Prolonging Wireless Sensor Networks Lifetime', *Wireless Networks*, Vol. 14, No. 6, pp.831–858.

Keskin, M. E., Altinel, I. K., Aras, N. and Ersoy, C. (2014) 'Wireless Sensor Network Lifetime Maximization by Optimal Sensor Deployment, Activity Scheduling, Data Routing and Sink Mobility', *Ad Hoc Networks*, Vol. 17, pp.18–36.

Yun, Y., Xia, Y., Behdani, B. and Smith, J. C. (2013) '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.

Wolsey, L.A. 1998 'Integer Programming', *Wiley-Interscience*

Basagni, S., Carosi, A., Petrioli, C. and Cynthia, P. (2011) 'Coordinated and controlled mobility of multiple sinks for maximizing the lifetime of wireless sensor networks', *Wireless Networks*, Vol. 17, No. 3, pp.759–778.

Sensor Network Museum, http://www.snm.ethz.ch (Accessed 25 February 2015)

Advanticsys SG 1000, http://www.advanticsys.com/ (Accessed 25 February 2015)

Koc, M. and Korpeoglu, I. (2014) 'Controlled Sink Mobility Algorithms for Wireless Sensor Networks', *International Journal of Distributed Sensor Networks*, Vol. 2014.

Kellerer, H., Pferschy, U. and Pisinger, D. 2004 'Knapsack Problems', *Springer*

Heinzelman, W., Chandrakasan, A. and Balakrishnan, H. (2000) 'Energy-Efficient Communication Protocol for Wireless Microsensor Networks', *33rd International Conference on System Sciences.*

Magazine, M. J. and Chern, M. (1984) 'A Note on Approximation Schemes for Multidimensional Knapsack Problems', *Mathematics of Operations Research*, Vol. 9, No. 2, pp.244–247.

IBM CPLEX Optimizer, http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/ (Accessed 25 February 2015)