Using Sink Mobility to Increase Wireless Sensor Networks Lifetime

Mirela Marta and Mihaela Cardei *
Department of Computer Science and Engineering
Florida Atlantic University
Boca Raton, FL 33431, USA
E-mail: {mmarta@, mihaela@cse.}fau.edu

Abstract

A critical issue for data gathering in wireless sensor networks is the formation of energy holes near the sinks. Sensors near the sinks have to participate in relaying data on behalf of other sensors and thus will deplete their energy very quickly, resulting in network partitioning and limitation of the network lifetime. The solution that we propose in this paper is to use mobile sinks that change their location when the nearby sensors' energy becomes low. In this way the sensors located near sinks change over time. In deciding a new location, a sink searches for zones with richer sensor energy.

First, we study the improvement in network lifetime when sinks move on a predetermined path, along the perimeter of a hexagonal tiling. Two cases are considered for data gathering when sinks stop in the hexagon's corners and when the sinks stop on multiple locations on the hexagon perimeter. This study shows an improvement of up to 4.86 *times in network lifetime. Second, we design a distributed and localized algorithm used by the sinks to decide their next movement location such that the virtual backbone formed by the sinks remains interconnected at all times. Simulation results are presented to verify our approaches.*

Keywords: Wireless sensor networks, energy-efficiency, data gathering, sink mobility, network lifetime.

1 Introduction and Related Work

This paper addresses the topic of energy efficient data gathering in wireless sensor networks (WSNs) consisting of sensor nodes deployed randomly in large number and several mobile sinks used to collect data from the sensors. We consider that the sinks have two transceivers, one to communicate with the sensors, and another to communicate with the other sinks. Sensors send their data to the sinks using multi-hop communication.

In WSNs, sensors closer to a sink tend to consume more energy than those farther away from the sinks. This is mainly because, besides transmitting their own packets, they forward packets on behalf of other sensors that are located farther away. As a result, the sensors closer to the sink will drain their energy resources first, resulting in holes in the WSN. This uneven energy consumption will reduce network lifetime.

WSN lifetime can be significantly improved if the energy spent in data relaying is reduced. One method to avoid formation of energy holes is to use sink mobility. When the energy of the sensors near a sink becomes low, the sink can move to a new location in a zone with richer sensor energy. This approach will balance the energy consumption and will increase network lifetime. Recent advances in the field of robotics [2, 5] make it possible to integrate robots as sinks (or gateways) in WSNs [11].

Adding mobile devices to WSNs infrastructure has attracted increased attention recently. Much of the work has been conducted on data gathering applications, where the mobile sinks move randomly ([10, 12]), using predetermined paths ([7, 11, 3]), or autonomously ([1]). A random moving sink is not aware of the sensor residual energy, and thus might threaten the energy balance among the sensors. The predetermined path models lack flexibility and scalability with network size. The moving strategies where the sinks take the moving decisions autonomously can better adapt to various network conditions.

In [7], the authors consider a WSN with a mobile base station which repeatedly relocates to change the bottleneck nodes closer to the base-station. Various predetermined trajectories are considered in the search for a combined optimum on the moving and routing strategies. Another approach is presented in [11], where authors consider a mobile base station which moves along a predetermined path. The sensor nodes are organized in clusters, where cluster-heads are the nodes closest to the mobile node trajectory. Cluster-

*This work was supported in part by NSF grant CCF 0545488.

heads are in charge of collecting data from their clusters and sending them to the mobile node when it passes by. A prototype using Mica 2 motes and the packbot mobile platform is developed and tested.

Another related work [3] explores the impact of predictable sink movement when the sink is mounted on a bus moving on a predictable schedule. Data is pulled by the sink which wakes up the nodes when it gets closer to them. This paper assumes that the sink comes within direct radio range of all the sensors during its movement, thus data is collected directly from the sensors. Using this approach, the power consumption for data relaying is saved.

Paper [1] considers a WSN with one mobile sink that moves proactively towards the node that has the highest residual energy in the network, in an effort to balance sensors' energy consumption. When the sink reaches a new location, it broadcasts a notification message and sensor data is collected using multi-hop communication.

In our paper, we consider a WSN consisting of a large population of sensors and multiple mobile sinks deployed for data gathering. We first study the improvement in network lifetime when sinks move on a predetermined path. Then we design a distributed and localized algorithm to determine sinks new locations in zones with richer sensor energy, while maintaining sinks connectivity.

Our work differs from the previous works by considering a multi sink design. We consider that the sinks form a virtual backbone and are concerned with maintaining the backbone connectivity as result of sinks' movements. In addition, compared to [1], when a sink moves, our algorithm searches for zones of sensors with high energy, not only the highest energy sensor.

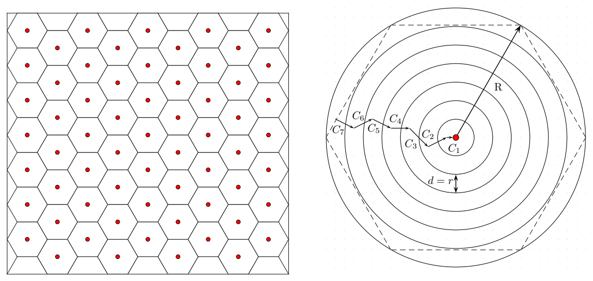
The remainder of this paper is organized as follows. We present the network model and the MS-NLI problem definition in section 2. We continue in section 3 with a study on the sinks mobility with pre-established trajectories. Section 4 presents the design of our distributed and localized algorithm for autonomous sinks movement. Section 5 presents our simulation results and section 6 concludes our paper.

2 Network Model and Problem Definition

2.1 Network Model

In this paper, we consider a heterogeneous WSN consisting of sensors and sinks. The sinks are deployed in the sensing area, they are connected, and their main task is to relay data from the sensor nodes to the user application. We make the following assumptions regarding the network model:

• We consider a periodic data gathering application where data is sensed and b data bits are transmitted by each sensor, in each time period T, to the closest sink. The data is forwarded to the sink using multihop communication.
• Sensor nodes are uniformly and randomly distributed.
• All sensor nodes have the same transmission range of r units. All sinks have the same transmission range of R units.
• Ideal MAC layer with no collisions and retransmissions.
• Each link has enough capacity to transfer the data.
• Sinks have movement capabilities.



(a) Sensor Network Organization for Static Sinks (b) Coronas Division for a Sink

Figure 1. Wireless Sensor Network Organization and Coronas Division

Paper [6] considers a similar network model for a WSN with only one static sink. The energy model of a sensor includes the power for sensing, power for receiving, and the power for transmission. The energy to sense, transmit, and receive b bits is computed as follows:

$E_{sense} = \alpha_1 b$
$E_{TX} = (\beta_1 + \beta_2 r^n)b$
$E_{RX} = \gamma_1 b$

According to [9], some typical values for the parameters are:

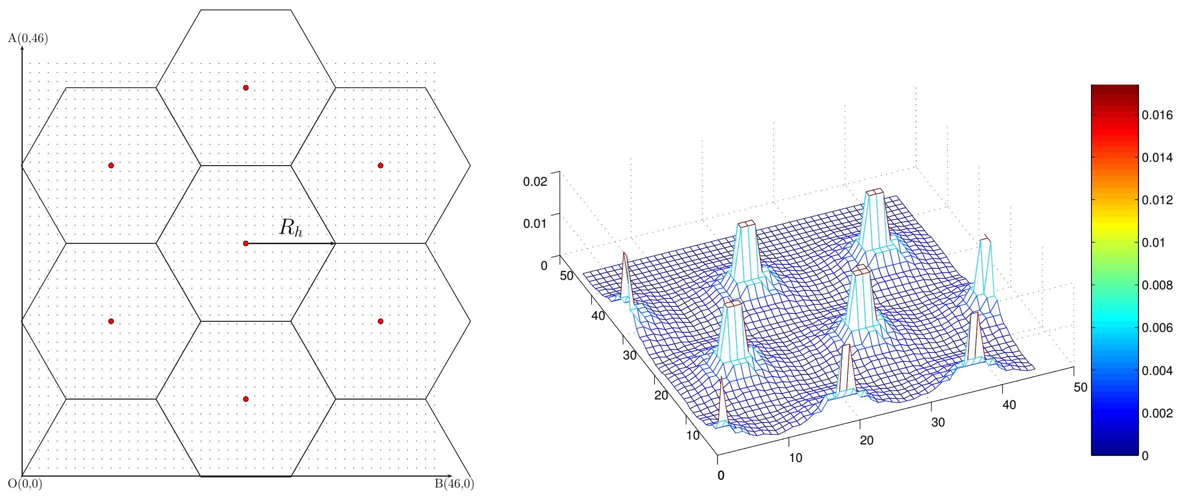
$\alpha_1 = 60 \times 10^{-9}$ J/bit
$\beta_1 = 45 \times 10^{-9}$ J/bit
$\beta_2 = 10 \times 10^{-12}$ J/bit/m^2, when $n = 2$
$\beta_2 = 0.001 \times 10^{-12}$ J/bit/m^4, when $n = 4$
$\gamma_1 = 135 \times 10^{-9}$ J/bit.

We consider that sinks have unlimited energy resources and thus we do not account the energy spent by the sinks. We define the network lifetime as the time interval until a sensor dies as result of depleting its energy resources.

In section 2.2, we discuss network lifetime for the case when sinks are static. This motivates our problem defined in section 2.3.

2.2 Network Lifetime using Static Sinks

For the case when the sinks are static, we consider that the sensing area is divided into hexagonal tiling, with the



(a) Multi-Sink Sensor Network Organization (b) Sensor Energy Consumption after T Time

Figure 2. Wireless Sensor Network with Static Sinks

sinks being located in the hexagon centers, as illustrated in the Figure 1.

Using this model, each sensor sends its data to the closest sink. In order to compute the energy consumed by each sensor, we use a corona-based model, similar to [6, 8]. The area around a sink is divided into coronas of width d, where $d = r$. A data message transmitted from corona C_i is sent to the closest sink using multihop communication, and it is forwarded by sensor nodes in coronas C_{i-1}, C_{i-2}, and so on until it reaches corona C_1 from where it is transmitted to the sink (see Figure 1b). Corona width is chosen such that a message is forwarded by only one sensor in each corona. As remarked in [6, 8], sensors suffer an uneven energy depletion, with sensors in the first corona being the first to die. This results in network partitioning, with other sensors being unable to report their data to the sink.

Let us denote with k the number of coronas and ρ the sensor density. The traffic load of a sensor includes both the data sensed by that sensor as well as the data forwarded on behalf of other sensors in higher coronas. The traffic load of a sensor in corona C_i, $i = 1 \ldots k$, is computed as follows:

$$Load_i = \frac{traffic\ from\ coronas\ C_i, C_{i+1}, \ldots, C_k}{number\ of\ sensors\ in\ C_i}$$
$$Load_i = \frac{\rho(\pi(kr)^2 - \pi((i-1)r)^2)b}{\rho(\pi(ir)^2 - \pi((i-1)r)^2)} = \frac{k^2-(i-1)^2}{i^2-(i-1)^2}b$$

It follows that the energy consumed by a sensor in corona C_i, $i = 1 \ldots k$, is computed as:

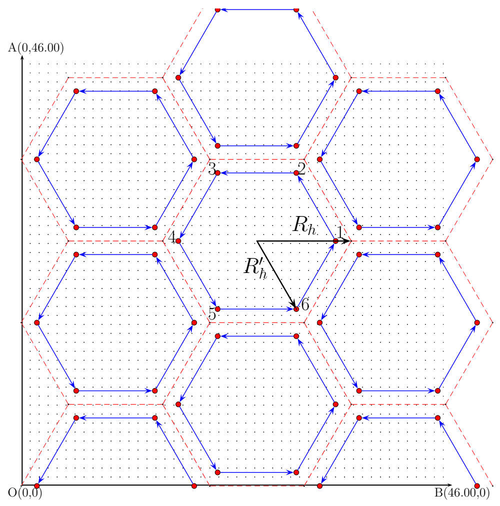
$$E_i = E_{sense} + E_{TX} + E_{RX} = \alpha_1 b + (\beta_1 + \beta_2 r^n)Load_i + \gamma_1 \frac{traffic\ from\ coronas\ C_{i+1}, \ldots, C_k}{number\ of\ sensors\ in\ C_i} = \alpha_1 b + (\beta_1 + \beta_2 r^n)\frac{k^2-(i-1)^2}{i^2-(i-1)^2}b + \gamma_1 \frac{k^2-i^2}{i^2-(i-1)^2}b$$

To show the difference in energy consumption between sensors in different coronas, we conducted a Matlab simulation for a heterogeneous WSN, using the parameters from Table 1. We measured the energy consumption of each sensor during a time period T.

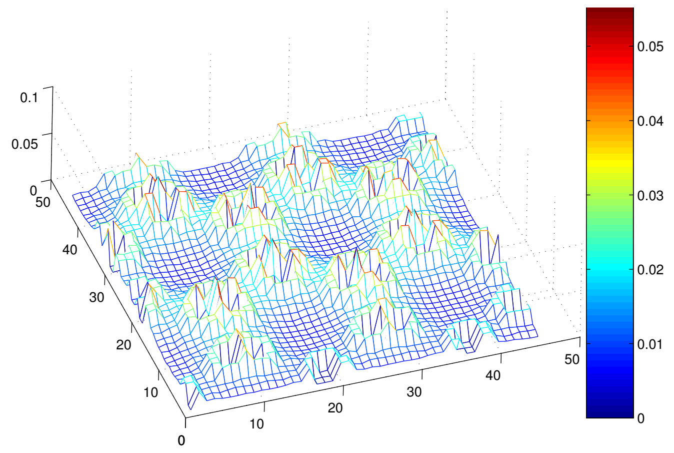
Symbol	Name	Value
A	Deployment area	46 × 46 square units
N	Number of sensors	$2116 = 46^2$
M	Number of sinks	7
R	Sink communication range	20 units
r	Sensor communication range	1.5 unit
R_h	Radius of the hexagon in the hexagon tiling	10.7 units
k	Number of coronas	7
b	Bits sent by each sensor in time T	2000 bits

Table 1. Simulation Parameters

As illustrated in the Figure 2b, the largest energy consumption takes place for the sensors in corona C_1. This happens because these sensors have the highest load, being involved in forwarding data generated from coronas $C_2, C_3, \ldots, C_k$. In our simulation results in Figure 2b, the energy consumed by the sensors in the seven coronas is: $E_1 = 0.0175$ J, $E_2 = 0.0056$ J, $E_3 = 0.0031$ J, $E_4 = 0.0019$ J, $E_5 = 0.0012$ J, $E_6 = 0.0006$ J, and $E_7 = 0.0002$ J. A sensor in corona C_1 consumes more than three times compared to a sensor in corona C_2. This result shows that network lifetime is limited by the sensors in



(a) Sinks Movement Trajectories



(b) Sensor Energy Consumption in Joules after $6T$ Time

Figure 3. Wireless Sensor Network with Mobile Sinks (6-Positions Sink Movement).

corona C_1 which will deplete their energy resources first, triggering network partitioning and impossibility to collect sensor data.

2.3 Problem Definition

The study in section 2.2 motivates our work on using sink mobility to increase network lifetime by balancing sensor energy consumption.

The problem of Sink Mobility for Network Lifetime Increase (SM-NLI) is formalized as follows: *Given a heterogeneous WSN consisting of N sensors randomly deployed for periodic monitoring of an area and M sinks with mobility capabilities, design a sink movement plan such that the network lifetime is maximized and the sinks remain interconnected all the time.*

The objective of the SM-NLI problem is to design the sinks movements to balance the sensors energy consumption and to vary the set of sensors located in first coronas. This will have a direct impact on network lifetime measured as the time until the first node depletes its energy resources. Another important requirement is to maintain the sinks interconnected. The sinks can be viewed as forming a *virtual sink backbone*, used for inter-sink communication and user data access.

In this paper, we propose two approaches for a sink movement path. In section 3, we address the case when sinks move on a predetermined path along a hexagon perimeter. In section 4 we propose a distributed and localized algorithm for sinks movements. Simulation results are presented in section 5.

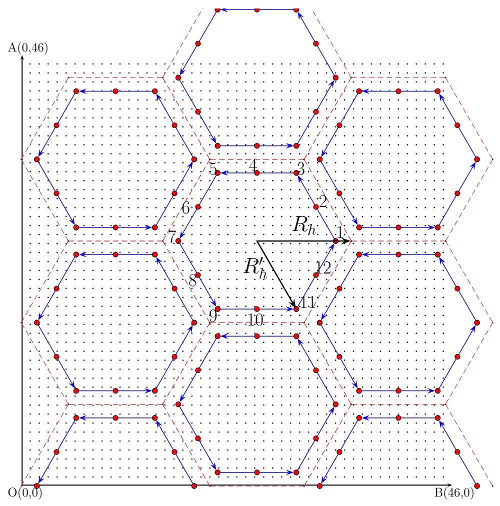
3 Sink Mobility with Pre-established Mobility Path

In this section we consider the case when sinks move along the hexagon perimeters. We assume that sinks' movements are synchronized, therefore the sinks relative positions remain the same at all the time. Assuming that the sink backbone was initially connected, it follows that the backbone remains connected at all times during the sinks movement. In our simulations, we took $R = 20$ units, resulting that each sink is connected with each of its six neighbor sinks.

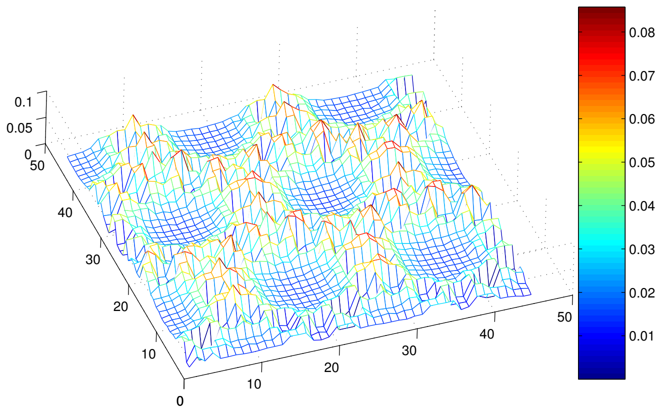
Each sink moves along the perimeter of a solid-line hexagon, as represented in the Figure 3a. The sink stops in the corners of the hexagon. In each stop, the sink collects data over a period T and then moves to the new location. After stopping in the six corners, the movement cycle is repeated. The movement positions of the middle sink are denoted with $1, \ldots, 6$ in the Figure 3a.

For a tiling with radius R_h (dotted-line hexagon), a sink moves in the corners of a hexagon (solid-line hexagon) with radius $R'_h = R_h - \frac{2}{\sqrt{3}}(d + \epsilon)$, where ϵ is a small, positive constant. R'_h was chosen such that the distance between the dotted-line hexagon and the solid-line hexagon is $d + \epsilon$. In this way, when sinks move along the solid-line hexagon perimeter, we prevent sensors nodes to belong to first corona of two different (neighbor) sinks.

We used the parameters in Table 1 for our simulation and we took the constant $\epsilon = 0.01$ units. In Figure 3b, we measure sensor energy consumption after one cycle, that means after a sinks gets back to the original location. Thus, these



(a) Sinks Movement Trajectories



(b) Sensor Energy Consumption in Joules after $12T$ Time

Figure 4. Wireless Sensor Network with Mobile Sinks (12-Positions Sink Movement).

results reflect energy consumption after $6T$ time. As illustrated in the figure, the maximum sensor energy consumption during $6T$ time is 0.0302 J. This is averaged to 0.00503 J energy consumption in time T. Therefore, the 6-positions sink movement resulted in an $0.0175/0.00503 = 3.48$ times improvement in network lifetime compared to the static sinks case.

The second simulation case that we considered is when a sink uses more positions in its movement along the solid-line hexagon, as illustrated in Figure 4a. In this case each sink moves in 12 positions. The figure shows the movement positions of the middle sink, denoted with $1, \ldots, 12$. Similar to the previous case, a sink collects sensor data over a period of time T and then moves to the new location.

The number of sink locations on the solid-line hexagon perimeter depends on the simulation parameters. Our goal is to use as many locations as possible while requiring that no sensor belongs to the first coronas for two different sink locations. First, we consider that each sink stops in the corners of its corresponding solid-line hexagon. Since the hexagon edge length is R'_h, a sink can stop in $l = \lfloor \frac{R'_h}{2d+\epsilon} - 1 \rfloor$ additional locations on each edge, such that the distance between two consecutive locations is greater than or equal to $2d + \epsilon$. Sink locations could be computed starting from the corners, at increment of $R'_h/(l+1)$. In our simulation $l = 1$, resulting in a total of $6 + 6 = 12$ sink locations.

We used the parameters in Table 1 for our simulation. Figure 4b measures the energy consumption after one cycle, that means after $12T$ time. As illustrated in Figure 4b, the maximum sensor energy consumption is 0.0425 J during a cycle of 12T time. This is averaged to 0.0036 energy consumption during time T. The energy consumption in this case is more balanced among all the sensors. This case results in 4.86 and 1.39 times improvement in network lifetime compared with the static sinks case and the 6-positions sink movement case, respectively.

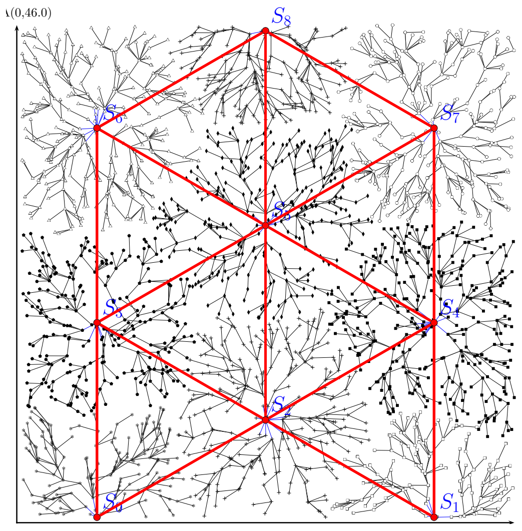
4 Sink Mobility with Unrestricted Mobility Path

In section 3 we showed the improvement in network lifetime when each sink follows a predetermined path. There are applications that assume that each sink can move autonomously, without following a predetermined trajectory. Such an example is when the sinks are robots or unmanned vehicles. Using a moving strategy where sinks take movement decisions autonomously can better adapt to various network conditions, environment conditions, and sensor deployment.

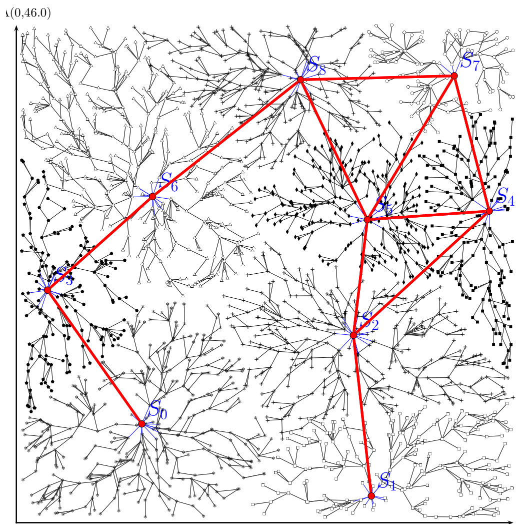
In this section we address the SM-NLI problem for the case when the sinks move autonomously such that (1) the sinks remain interconnected all the time forming a *virtual sink backbone*, and (2) network lifetime is maximized.

We consider that the data gathering mechanism is organized in rounds of time T. At the beginning of each round, data collection trees are established using a clustering mechanism. Each sink serves as a cluster head and it broadcasts a *CLUSTER_INIT (ID, hops=0)* message containing the sink id and the number of hops which is initially zero. Each sensor node maintains information about the closest sink and forwards only messages from which it learns about a closer sink:

```
1: min_hops = $\infty$; cluster_id = NIL;
```



(a) Initial Virtual Sink Backbone



(b) Virtual Sink Backbone after 20 Rounds

Figure 5. Example of the Distributed Algorithm with 2116 Sensors and 9 Sinks.

```
2: if CLUSTER_INIT(ID, hops) message received then
3:    if hops+1 $<$ min_hops then
4:       cluster_id = ID
5:       min_hops = hops+1
6:       next_hop = sensor from which this message was received
7:       rebroadcast the message CLUSTER_INIT(ID, hops+1)
8:    end if
9: end if
```

Once the clusters have been constructed, sensor data are collected along the paths formed by the next_hop field.

At the end of each round, a sink decides whether or not it moves to a new location, depending on the energy levels of its 1-hop sensor neighbors. These are the sensor nodes that will deplete their energy first since they also have to forward messages on behalf of other sensors.

Figure 5a shows the initial deployment of an example with 2116 sensors and 9 sinks. Sensors in different clusters are represented using different symbols. Figure 5b shows the virtual backbone after 20 rounds. Sinks move in zones with higher energy sensor nodes, but they remain interconnected at all times. The *Decide-Sink-Movement* algorithm which is executed by a sink S_i is presented next:

Decide-Sink-Movement($S_i, p, q, E_{th}, E'_{th}$)

```
1: if $p\%$ of the 1-hop sensors have $E \leq E_{th}$ then
2:    $\triangleright$ sink $S_i$ searches for a new location
3:    new-location = Find-Best-Location($S_i$, $E'_{th}$)
4:    if new-location $\neq \emptyset$ then
5:       sink $S_i$ moves to the new-location
6:    else if $E_{th} \geq E_{min}$ then
7:       $E_{th} = \beta \cdot E_{th}$
8:       go to line 1
9:    else
10:      sink does not move
11:   end if
12: end if
```

We consider that the sensors 1-hop away from the sink send their current energy levels to the sink at the end of each reporting interval. This information can be piggybacked to a data message. If at lest $p\%$ of the sensors have reached the low threshold energy E_{th}, then the sink searches for a new zone where sensors have richer energy resources, using the algorithm *Find-Best-Location*. The zone where the sink moves must have energy at least E'_{th}, where $E'_{th} = E_{th} + \alpha \cdot E_{th}$ and $0 < \alpha < 1$. For example, when $\alpha = 0.1$ ($\alpha = 10\%$), then $E'_{th} = 1.1E_{th}$.

If a new location is found, then the sink moves to that location. If no new location is found, then the overall energy level of the nodes has decreased, thus the energy level E_{th} adjusts dynamically to a smaller value, $E_{th} = \beta \cdot E_{th}$, where $0 < \beta < 1$. For example, when $\beta = 0.75$ ($\beta = 75\%$), then $E_{th} = 0.75 \cdot E_{th}$. In this case, first the sink waits for $p\%$ of the 1-hop sensors to have $E \leq E_{th}$, then a new location is searched. The energy threshold is adaptively adjusted until it becomes lower than E_{min}, and after that it will not change. The algorithm *Find-Best-Location(S_i)* is presented next.

Find-Best-Location(S_i, E'_{th})

```
1:  $n_{hops}$ = initial value
2:  while $n_{hops} \leq n_{max}$ do
3:    $S_i$ broadcasts LOCATION-REQ($S_i$-ID, req-ID, $n_{hops}$, $E'_{th}$)
4:    wait some specific time and record all sensor reply messages LOCATION-REPLY($s_k$, $\#msg$, hops)
5:    if one or more LOCATION-REPLY messages are received then
6:      sort messages in decreasing order of $\#msg$ values
        let LOCATION-REPLY($s_k$, $\#msg$, hops) be the first message in the sorted order
7:      if Connected-Backbone($S_i$, $s_k$) then
8:        return new-location = $s_k$
9:      else
10:       take the next message in the sorted order and go to line 7
          if no more messages, go to line 13
11:     end if
12:   end if
13:   increment $n_{hops}$
14: end while
15: return $\emptyset$
```

Sink S_i uses an incremental ring approach to search for a new location. We use as candidate sink locations the sensor locations in S_i's cluster. We use this approach since sensors are densely deployed and a sensor can easily check the energy level information of its 1-hop neighbors. The number of hops n_{hops} is set-up initially to a small value, since the closer the new location is, the smaller the sink movement distance is. If no valid location is found for the sink, the number of hops n_{hops} is incremented (in line 13) until eventually the whole cluster is reached.

For a specific n_{hops} value, sink S_i broadcasts a *LOCATION-REQ* message in its n_{hops}-neighborhood. Each sensor receiving the message decrements n_{hops} value and forwards the message along the cluster tree. Each sensor will temporary store the (S_i*-ID*, *req-ID*) values of the most recent *LOCATION-REQ* message received. When a sensor s_k receives a *LOCATION-REQ* message, it checks the (S_i*-ID*, *req-ID*) values. If they are the same as the values s_k has stored, then no action is taken other than forwarding the message. Otherwise, s_k exchanges *HELLO* messages with its 1-hop sensor neighbors, containing the current energy levels. If all s_k's neighbors have the energy at least E'_{th}, then s_k is a candidate location for the sink and thus it sends a *LOCATION-REPLY(*s_k*,* $\#msg$*, hops)* message back to the sink. *hops* is the number of hops between s_k and S_i, and $\#msg$ represents the number of messages transmitted by the node s_k in one round.

After the sink S_i has sent the request, it waits a specific time for sensor replies. This waiting time is proportional to the size of the search neighborhood, characterized by n_{hops}. After the waiting time has expired, the sink sorts the received *LOCATION-REPLY* messages in decreasing order of the $\#msg$ field. The sink gives priority to the sensor locations that forward a large number of messages since these sensors deplete their energy at the fastest rate.

The sink considers the *LOCATION-REPLY* messages in the decreasing order of the $\#msg$ field and checks if the candidate location satisfies the sink backbone connectivity requirement. This algorithm is presented in the *Connected-Backbone* procedure. If the sink connectivity is satisfied then the sink S_i returns s_k's location as its new location.

If no candidate location is valid due to the connectivity requirement, then S_i increments n_{hops} in order to increase the search neighborhood. When a new *LOCATION-REQ* message is sent by S_i, the sensors which have received the previous request message, and thus have stored the (S_i*-ID*, *req-ID*) values will not compete as candidate locations. They only participate in message forwarding.

If, after the whole cluster has been searched, no candidate location has been found, then the sink does not move to a new location. The algorithm to determine if a new sink location maintains backbone connectivity is presented next.

Connected-Backbone(S_i, s_k)

```
1: $S_i$ computes its $l$-hop sink neighborhood, denoted $\Gamma(S_i)$
2: Construct a graph $G$ with one vertex for each sink in $\Gamma(S_i)$. Add one more vertex for $S_i$'s tentative new location $s_k$. Add an edge between every two vertices if their distance is at most $R$.
3: Run BFS(G) to check graph connectivity
4: if $G$ is connected then
5:   return TRUE
6: else
7:   return FALSE
8: end if
```

We assume that each sink has two transceivers, one for communication with sensor nodes and the other for communication with other sinks. A sink communication range is denoted R. A sink S_i determines its l-hop sink neighborhood $\Gamma(S_i)$, by exchanging *HELLO* messages with its sink neighbors with $TTL = l$. We take l an input parameter that usually has small values (e.g. $l = 1$ or $l = 2$). We assume that each sink knows its current location and that this information is included in the *HELLO* messages.

Sink S_i then constructs the undirected graph G where one vertex is allocated for every sink in $\Gamma(S_i)$ and one vertex is allocated for S_i's tentative location s_k. Edges are added between any two vertices if the corresponding sinks locations are at distance less than or equal to R.

To determine if S_i is still connected to the graph G in the new location, the sink S_i runs the Breadth-First-Search (BFS) algorithm [4] for the graph G. If the resultant BFS tree is connected, then the algorithm returns TRUE, otherwise it returns FALSE.

Another issue that we need to consider is avoiding multiple neighbor sinks to move simultaneously. For this, we can use a locking mechanism. If BSF tree is connected, the sink sends a *LOCK* message to its $\Gamma(S_i)$ neighbors. Then, after the sink S_i moves to the new location, it sends an *UNLOCK* message to the sinks in $\Gamma(S_i)$. The *UNLOCK* message will also contain S_i's new location.

The algorithm run by a sink to decide its movement is a localized algorithm. In a local solution, a decision at each node is based on local information, without any information propagation in the whole network. Localized algorithms are important for WSNs, being scalable with the number of nodes in the network. In the *Decide-Sink-Movement* algorithm, the sink movement decision is based on the energy level of its 1-hop sensor neighbors. To find the best location to move, a sink considers locations in its own cluster. The closer locations are considered first, using an incremental ring search approach. To check if the new location satisfies the sink connectivity requirement, the *Connected-Backbone* algorithm is invoked. This is a localized algorithm where the sink uses its l-hop neighbor information.

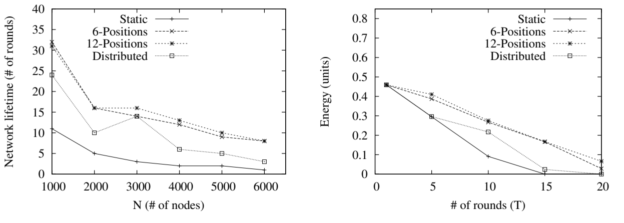
5 Simulation

In this section, we analyze and compare the performance of the three algorithms that we have presented in the paper for the SM-NLI problem, versus the static case. In summary, we consider the following sink movement decision algorithms:

1. Static, from section 2.2, where sinks do not move.
2. 6-Positions, the algorithm from section 3, where each sink moves along the perimeter of a hexagon, stopping in the hexagon's corners.
3. 12-Positions, the algorithm from section 3, where each sink stops in 12 positions, following the perimeter of a hexagon.
4. Distributed, the algorithm from section 4.

We have simulated the algorithms using a custom Java application. In order to compare the four algorithms, we have used the energy model described in section 2.1. In all four simulated algorithms, data gathering is performed using the clustering mechanism described in section 4, where sinks serve as cluster-heads.

The sensors are randomly deployed in a 46×46 area units. All sensor nodes have the same capabilities. In our simulations, we have considered the cases when sensors are deployed using a random uniform distribution and a bivariate Gaussian distribution. The sinks are uniformly distributed in a hexagonal tiling. At the network start-up, each sensor has an initial energy of 0.5 units. Unless otherwise specified, we used the following values for the distributed algorithm: $p = 1$, $E_{th} = 0.4$, $\alpha = 0.05$, and $\beta = 0.75$. In our simulations we measure only the energy consumed on sensing, data transmission, and data reception. We do not account the energy spent by sensors during the sinks relocation mechanism.



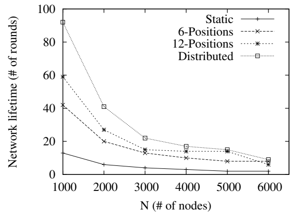
(a) Network Lifetime (b) Power Consumption per Rounds

Figure 6. Network lifetime and power consumption per round for all algorithms when sensors are deployed using a random uniform distribution.

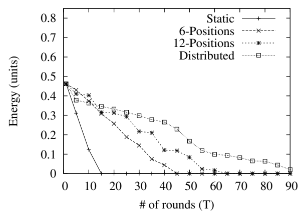
In the simulations we vary the following parameters:

1. The network size N is varied to examine the scalability of the network. The size of the network is varied between 1000 and 6000.
2. The number of sinks M is varied between 1 and 9.
3. The data aggregation factor is varied between 0 and 0.75 to analyze network lifetime in the distributed case.
4. Parameter p in the distributed algorithm takes the values: 1, 5%, 10%, 15%, and 30%. Parameter p has the following meaning: when p percentage of the sink's 1-hop sensors have reached the low energy threshold, the sink moves to a new location. In the case $p = 1$, the sink moves when at least one sensor has reached the low energy threshold.
5. The sensor communication range r takes values between 1 and 4.

For each tunable parameter, each simulation was repeated 10 times and the results averaged. Network activity is organized in rounds, with each sensor sending one packet in each round. At the end of each round, every sink decides if it moves to a new location based on its movement decision criteria. The main performance metric is the network lifetime. We define the network lifetime as the number of rounds until the first sensor node dies.

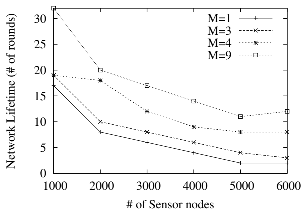


(a) Network Lifetime

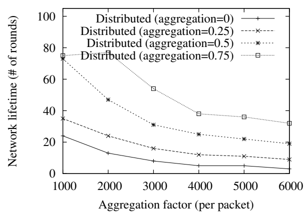


(b) Power Consumption per Rounds

Figure 7. Network lifetime and power consumption per round for all algorithms when sensors are deployed using a bivariate Gaussian distribution.



(a) Network Lifetime for Different Number of Sinks



(b) Network Lifetime for Different Aggregation Factor

Figure 8. Network lifetime when we vary the number of sinks and the aggregation factor.

In the simulations we have used sensor communication range $r = 2$ units and sink communication range $R = 20$ units. In all figures, except Figure 8a, the topology contains 9 sinks which are uniformly distributed using a hexagonal tiling, similar to the sink deployment in Figure 2a.

Figure 6a compares the network lifetime of the three algorithms versus the Static case. Sensors are deployed using a random uniform distribution. All the three algorithms that employ sink movement perform better than the static case, when the sinks do not move. The best results are obtained for the cases when the sinks move on the hexagonal, predetermined trajectory. The distributed algorithm is a localized solution and it obtains a shorter network lifetime than the predetermined trajectory cases. In that case, the sinks movement is synchronized, thus the data collection trees are more balanced. Figure 6b shows the variation of the minimum sensor energy over time for a network with 1000 sensors. The minimum sensor energy after 1 to 20 rounds is plotted. This result is consistent to Figure 6a, with the static case reducing the energy most abruptly.

Figure 7a compares the algorithms' performance for the case when sensors are deployed using a bivariate Gaussian distribution. Consistent with the previous results, the cases when the sinks move result in an improved network lifetime. The distributed algorithm has the best performance, followed by the predetermined-paths cases. This graph shows the benefit of using a distributed algorithm which adapts with the deployed network topology.

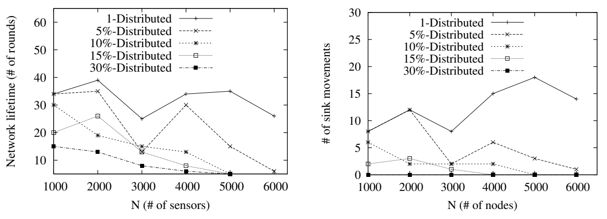
The sinks in the 6-Positions and 12-Positions mechanisms move synchronized and they follow the hexagonal trajectories. In this case, the sinks do not partition the sensor network as regularly as it was for the simulations in Figure 6. Even if a node is the root of a large (or small) data collection tree, the sink will visit that position regularly. Such a situation is avoided in the distributed algorithm, where sinks move when 1-hop neighbors are energy-depleted. Figure 7b shows the minimum sensor energy for a network with 1000 sensors, when the number of rounds vary between 1 and 90.

The next graphs, Figures 8 to 10, show the simulation results for the distributed algorithm with random uniform sensor deployment, when we vary different parameters. Figure 8a compares network lifetime when we vary the number of sinks between 1 and 9. We can observe that a larger number of sinks helps in improving network lifetime. A larger number of sinks results in more clusters, with smaller data collection trees.

All the previous simulations assume that no data aggregation is performed. In Figure 8b we study network lifetime for different aggregation factors $f = 0, 0.25, 0.5$, and 0.75. The aggregation is done at the packet level. An aggregation factor f means that $(1 - f) \times 100\%$ of the packets are forwarded by a sensor. Thus, an aggregation factor $f = 0$, means that no aggregation is used. As illustrated in the graph, the larger the aggregation factor, the larger the network lifetime is.

In Figure 9, we vary the parameter p in the distributed algorithm, when $M = 9$ sinks. In the distributed algorithm, if at least $p\%$ of the 1-hop sensors of a sink have reached the low energy theshold E_{th}, then the sink moves to a new location. 1-Distributed is the case when the sink moves if at least one 1-hop sensor has reached the low energy threshold. In Figure 9a, the largest network lifetime is obtained for 1-Distributed. In general, a larger number of rounds is obtained for smaller p. As illustrated in the Figure 9b, the trade-off is an increase in the number of sink moves for the 1-Distributed. Figure 9b represents the average number of sink movements in the network.

Figure 10 measures network lifetime when we vary the sensor communication range r between 1 and 4. In general, $r = 2$ and $r = 3$ produce better results. A small communication range will increase the number of hops that a message has to travel and thus will increase the number of forwarded messages. On the other hand, a large communication range reduces the number of forwarded messages but increases the energy spent on message transmission.



(a) Network Lifetime for Different Values of p (b) Average Number of Sink Movements for Different Values of p

Figure 9. Network lifetime and number of sink movements when we vary the parameter p in the distributed algorithm.

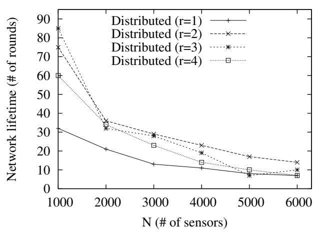


Figure 10. Network lifetime for different sensor communication range r.

The simulation results can be summarized as follows:

- Using sink mobility provides an effective mechanism to prolong network lifetime. When the sensors are deployed using a random uniform distribution, then a predetermined trajectory (6-Positions and 12-Positions) produces the best results. When the sensors are deployed using a Gaussian distribution, then the distributed algorithm produces the largest number of rounds.
- A larger number of sinks results in an increased network lifetime.
- Data aggregation is another effective method to increase network lifetime.
- Various selections of the parameter p can trade-off network lifetime with the number of sink movements.

6 Conclusions

In this paper we have studied the effect of using mobile sinks for data gathering in wireless sensors networks. If the sinks are static, the sensors near the sinks will deplete their energy first, resulting in an early disconnection of the network. One method to alleviate this problem and to obtain a more balanced energy consume is to use mobile sinks. We have first studied the improvement in network lifetime considering a hexagonal tiling, where sinks move along the hexagon perimeter. Our simulation study shows an improvement by 4.86 times in network lifetime compared with the static sinks case.

We have also proposed a distributed and localized solution to decide sinks movements when the movement path is not predetermined. When the sensors closed to sink have scarce energy resources, sinks moves to zones where sensors have richer energy resources and forward a large number of messages. Simulations results show significant improvements in network lifetime compared to the static sinks case.

References

[1] Y. Bi, L. Sun, J. Ma, N. Li, I. A. Khan, and C. Chen, HUMS: An Autonomous Moving Strategy for Mobile Sinks in Data-Gathering Sensor Networks, *Eurasip*, 2007.

[2] J. Butler, Robotics and Microelectronics: Mobile Robots as Gateways into Wireless Sensor Networks, *Technology@Intel Magazine*, May 2003.

[3] A. Chakrabarti, A. Sabharwal, and B. Aazhang, Using Predictable Observer Mobility for Power Efficient Design of Sensor Networks, in Proc. of the 2nd IEEE IPSN, 2003.

[4] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, Introduction to Algorithms, 2nd edition, Mc Graw Hill, 2001.

[5] A. LaMarca, W. Brunette, D. Koizumi, M. Lease, S. Sigurdsson, K. Sikorski, D. Fox, and G. Borriello, Making Sensor Networks Practical with Robots, in LNCS, F. Mattern and M. Naghshineh (Eds.), pp. 152-166, Springer-Verlag, 2002.

[6] J. Li and P. Mohapatra, An Analytical Model for the Energy Hole Problem in Many-to-One Sensor Networks, *Vehicular Technology Conference*, 2005.

[7] J. Luo and J.-P. Hubaux, Joint Mobility and Routing for Lifetime Elongation in Wireless Sensor Networks, *IEEE INFOCOM*, 2005.

[8] S. Olariu and I. Stojmenovic, Design Guidelines for Maximizing Lifetime and Avoiding Energy Holes in Sensor Networks with Uniform Distribution and Uniform Reporting, *IEEE INFOCOM*, 2006.

[9] W. Heinzelman, Application-Specific Protocol Architectures for Wireless Networks, Ph.D. thesis, Massachusetts Institute of Technology, 2000.

[10] R. C. Shah, S. Roy, S. Jain, and W. Brunette, Data MULEs: Modeling a Three-tier Architecture for Sparse Sensor Networks, *Proc. of the 1st IEEE Intl. Workshop on Sensor Network Protocols and Applications (SNPA'03)*, May 2003.

[11] A. A. Somasundara, A. Kansal, D. D. Jea, D. Estrin, and M. B. Srivastava, Controllably Mobile Infrastructure for Low Energy Embedded Networks, *IEEE Transaction on Mobile Computing*, Aug. 2006.

[12] L. Tong, Q. Zhao, and S. Adireddy, Sensor networks with mobile agents, *Proc. of IEEE Military Communications Conference (MILCOM'03)*, Oct. 2003.