Fast Sink Placement for Gossip-based Wireless Sensor Networks

Milos Blagojevic[1,2], Marc Geilen[1], Twan Basten[1,2], Teun Hendriks[2]

[1] *Eindhoven University of Technology, the Netherlands*

[2] *Embedded Systems Institute, Eindhoven, the Netherlands*

mblagojevic@tue.nl, m.c.w.geilen@tue.nl, a.a.basten@tue.nl, teun.hendriks@esi.nl

***Abstract*— In this paper we address the problem of sink placement for Gossip-based Wireless Sensor Networks (GWSN). Sink placement plays an important role in planning and deployment of sensor networks. It is an efficient means to improve performance and achieve design objectives. Sink deployment requires an optimization strategy to search a space of possible placement options, and a performance evaluation method to assess the quality of different sink placements. The stochastic nature of the gossip protocol makes this task challenging for GWSN. Simulation is the most common way to accurately evaluate gossiping performance; however, the time required to obtain statistically significant results is considerable and limits the scalability of the sink deployment process. We use a fast and accurate performance evaluation technique, which exploits specifics of the sink placement problem and significantly reduces evaluation time. In order to further improve the speed of the sink placement procedure we propose a greedy simulated annealing search heuristic that converges fast to a near-optimal placement. We have performed an extensive set of experiments to evaluate the performance of the proposed sink placement framework.**

Keywords- wireless sensor networks, sink placement, gossip protocols, optimization, performance evaluation, system modeling, stochastic shortest path, Monte Carlo simulation

I. INTRODUCTION

Wireless Sensor Networks (WSN) are complex systems of spatially distributed sensor nodes that communicate wirelessly, operate autonomously and perform some cooperative action. In this paper, we provide a solution to the problem of optimal deployment of multiple sink nodes in a gossip-based wireless sensor network (GWSN). In many applications, the information is extracted from the WSN at one or more special nodes, the sinks. Typically, a sink node has some additional resources to store data and perform application tasks. For example in environment monitoring applications a sink node may actuate various devices (alarms, lights, heating, etc.) based on the collected measurements. Typically, due to additional resources and available features, sink nodes have a higher cost.

The time required to deliver information to a sink node, latency, often plays an important role, e.g., in detection systems (fire detection, health monitoring [16]). Placing a sink node at an optimal position can improve latency noticeably. Latency can further be improved by placing multiple sinks in the network, at optimal positions. In that case the sink closest to the place of measurement, takes responsibility for collection and processing of the measured data. The achieved latency depends on both the number and the positions of the sink nodes. The number of sink nodes in the network represents a trade-off between latency and cost.

GWSNs are complex systems of inherently random nature [3],[4]. Gossip protocols rely on a probabilistic mechanism for disseminating data throughout the network. The system properties (e.g., latency) are the result of numerous interactions between large numbers of stochastic processes and evaluating them is a known performance evaluation challenge [2][3][4]. The main advantage of the gossip protocol is its robustness to node and link failures. Data is disseminated in multiple directions simultaneously, and single failures do not prohibit data delivery. In deployments with multiple sink nodes, if some sink nodes fail, the other sink nodes can take over their tasks. The price paid for this robustness is increased communication redundancy, and thus lower performance. An optimal placement of multiple sink nodes can improve performance for instance by shortening the maximal latency.

The goal of a designer is to find a (near-)optimal deployment of the WSN in a reasonably short time. The performance metrics corresponding to the application requirements (e.g., latency and cost) and the set of configuration parameters that have an effect on those metrics (e.g., the number of sink nodes and their positions) create the WSN design space. The size of the design space is huge, and exhaustive exploration is typically not an option. For example, even in a simple case where 5 sink nodes have to be deployed over 100 candidate locations the total number of possible deployments is of the order 10^{10}. The size of the design space increases exponentially with the number of candidate locations and sink nodes. Moreover, certain configuration parameters (such as transmission power and duty cycle) affect the connectivity properties of the network. In that case, the optimal sink placement for one configuration typically is not optimal for another configuration. In a large-scale design-space exploration the sink placement procedure may have to be performed many times, and should provide near-optimal solutions in a tight time budget.

In this paper, we present an integral framework for sink-placement optimization (SPO) in GWSN (Figure 1). The SPO framework integrates a fast performance evaluation technique and an iterative greedy simulated annealing (GSA) search heuristic. The greedy search strategy finds near-optimal sink positions in a small number of search iterations. The performance of the greedy search is affected by the initial sink placement. We propose a simple heuristic for initial placement as well. The simulated annealing approach extends the search scope of a pure greedy heuristic and

improves the quality of obtained solutions. For the purpose of the performance evaluation we use our Stochastic Variable Graph Model (SVGM) presented in [4], exploiting the specifics of the sink placement problem.

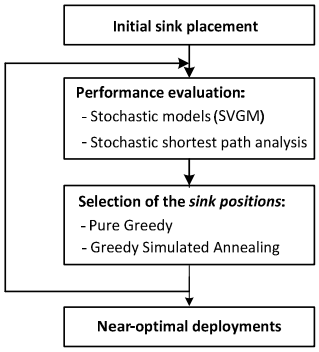

Figure 1. Sink-placement optimization framework

The next section gives an overview of related work. In Section III, a case study is introduced. Section IV introduces our method for performance evaluation. The performance evaluation results are used to guide an efficient optimization strategy described in Section V. Finally, Section VI gives simulation results and analyzes different aspects of the proposed SPO framework. Section VII concludes.

II. Related Work

Recent surveys [23] show that one of the main obstacles for wide spread use of WSN is the lack of installation ease; support for design space exploration and optimization is limited and efficient methods to deploy WSN in a predictable and controllable way are still missing. Planning and designing a network is an iterative process, consisting of several phases where various decisions about topology and network design have to be made. The optimal placement of one or more sink nodes is one of the phases often encountered in both wired and wireless networks. A thorough survey of strategies and techniques for sink placement in WSN is given in [27]. The quality of the sink placement can be evaluated according to different Quality of Service metrics, such as: energy [25][9], coverage [24], latency [28], etc. The framework proposed in this paper solves the problem of the latency-optimal placement of multiple sink nodes in gossip based WSN.

In order to deal with problem complexity, optimization methods typically reduce a search space. For instance, in [13] the search space is restricted to candidate grid positions, while in [11] the search space is limited to the sensor locations. In this paper, we use the latter approach and allow that sinks are positioned only at the locations where the sensor nodes are already deployed. The method can be adapted to the other types of candidate positions though.

Typically, even reduced, the search space is still large and a smart technique for its exploration is required. Genetic algorithms are a widely applicable tool giving robust solutions for different types of design-space optimization problems in WSN [20],[45], including SPO problems [28],[22]. A genetic algorithm starts with a group of random solutions (population) and improves the population of candidate solutions in an iterative process that mimics the process of natural evolution. A genetic algorithm provides a broad search scope, but its convergence is too slow for large-scale DSE. In contrast, a greedy approach converges fast, achieving a local optimum. Although its search scope is smaller than the scope of genetic algorithms, it is the preferred solution in all scenarios where the time budget for optimization is severely limited. The greedy SPO strategy presented in [6] deploys sink nodes incrementally, placing each sink at the best possible location considering only the sink nodes already placed (but not those that still have to be placed). Alternatively, some strategies [6] deploy sink nodes randomly and then perform a greedy search around those random positions. In order to provide a good solution, the latter approach requires that the search is repeated for several different random start positions. In this paper we develop a greedy search heuristic with smart initial deployment, which provides fast convergence towards an optimal deployment.

Greedy search makes deterministic decisions and chooses the best solution in each step of an iterative process. Simulated annealing [17] is a technique that in each search iteration selects with a certain probability some solutions that are not optimal. The probability that a non-optimal solution is selected decreases over time and eventually the search becomes pure greedy. The randomness in the search process allows avoiding low quality local optima. We integrate the annealing principle in our heuristic. Many SPO heuristics use structural quality metrics [27] (e.g., distance, hop-count, connectivity) to assess the quality of solutions and guide the search. Such approaches lead to good solutions when strong correlations between structural metrics and QoS metrics exist; e.g., for a shortest path routing, the (weighted) hop-count of the path between source and sink is a good indication of the latency. However, in case of more complex dissemination strategies correlation between structural metrics and latency is not that strong (see e.g., [21]). For example, the stochastic aspect of the gossiping dissemination is not captured by weighted hop-count approximation. Accurate evaluation of QoS metrics in GWSN requires stochastic analysis. Such approach achieves better accuracy at the cost of increased evaluation time. The overall effect of the performance evaluation accuracy on the search results is analyzed in Section VI.B.

The local behavior of a single node in a GWSN is easy to model, but the complex interactions between a large numbers of nodes with stochastic behavior are very hard to model. It is therefore not surprising that the dominant technique for performance evaluation for GWSN is simulation. Most alternative methods either cannot cope with system complexity (typically analytic models [2],[3]) or bring a lot of computation overhead (model-checking approaches [20],[12],[14]). Discrete-event simulators are the most popular method for GWSN analysis. Often, simulations adapt some of the general network simulators (ns-2 [30], Omnet++ [31], Opnet [32]) to the WSN specifics. The speed of such an approach is acceptable for analyzing system behavior and gaining insight in the interaction between system parts, but it is not sufficiently fast for SPO. In [4] we

propose a gossip-tailored performance evaluation method which uses probabilistic models of the system layers and integrates them into Stochastic Variable Graph Models (SVGM). The method combines dynamic programming (shortest-path analysis) and Monte Carlo sampling, allowing short evaluation times and good accuracy. In this paper, we adapt these ideas to the specifics of the SPO problem.

III. Motivating Case and Problem Definition

The proposed optimization framework for sink placement in GWSN is introduced through a motivating case study.

A. System description

We consider a static network deployment with nodes deployed at known positions, i.e., node positions have been determined by an application and a set of phenomena that have to be monitored. Each sensor in a regular period performs a measurement of the corresponding physical phenomena. The measurements are disseminated and stored in the form of data items. A data item is a piece of information describing the measured value, time-stamp (when the measurement was taken) and location of the measurement. Through gossiping, data items are delivered to each node in the network, including sink nodes. Each node has its own information database, called the cache, where data items are stored. We use a MyriaNed WSN platform [26], and its proprietary MAC and gossip protocols.

Communication is controlled by a TDMA-based MAC protocol, gMAC [1]. In each periodic communication round, a node can transmit only a small and constant number of data items. Nodes transmit packets in collision-free TDMA slots (unique in 2-hop neighborhoods and determined by gMAC). gMAC provides probabilistic TDMA scheduling, i.e., when a node transmits data items, with a certain probability each of the neighbor nodes decides whether to listen in that time slot or to remain in sleep mode.

The task of the gossiping protocol is to select data items from the node's cache to be transmitted in each communication round. The selection process can be modeled with the probability that a data item is selected from the cache [4]. This probability is one of the stochastic aspects used in the SVGM to evaluate the properties of the gossip based dissemination. For the sake of simplicity and without loss of generality we assume that measurements from all sensors have the same priority and that only the most recent version of each measurement present in the cache is considered in the selection process. The combination of gMAC and gossiping results in a system that disseminates the same data item simultaneously into multiple directions.

In this paper, we assume that sink nodes can be placed only at the locations where the sensor nodes are placed (determined by the application). The sink nodes then replace regular sensor nodes in the topology and have the additional responsibility to collect data from the network.

B. Latency

One of the important QoS metrics describing the performance of GWSN is latency, the average time required to disseminate information to the sink node.

Communication between nodes in GWSN is intrinsically unreliable, due to the stochastic nature of the radio channel, the data-link layer and gossiping. The time necessary for a node to spread information to its neighbors is therefore not fixed, but a stochastic variable that depends on the system attributes. Data items are simultaneously propagated in multiple directions, and multiple copies of the same data item are present at different locations in the network. The latency to reach some node is influenced by the distribution path (chain of nodes) that the item follows until the first time a node receives it. Figure 2 shows two examples of possible distribution paths for a data item from the bottom left node, eventually reaching all other nodes. An arrow between two nodes indicates that the item reached the arrow's sink node for the first time by a transmission from the arrow's source node. Both distribution trees are samples of the same stochastic process; due to stochastic variations in link delays the concrete paths turned out differently.

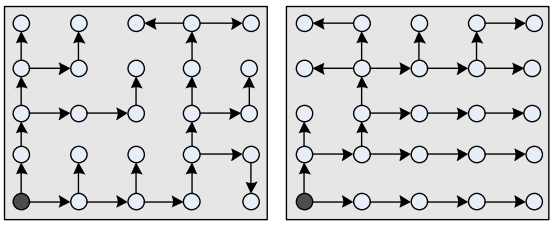

Figure 2. Distribution paths for data items generated at the lower left node.

The *point-to-sink latency* metric $L(k,s)$ denotes the average time necessary to deliver a data item measured at sensor node k to the sink node s. It is defined as the average time passed between the moment when a data item is generated at node k and the moment it appears in the cache of sink node s.

C. Optimization objective

Let K be the set of GWSN nodes with $S \subseteq K$ the set of sinks. The *delivery latency* of a data item generated at node k, L_k, is the point-to-sink latency to the *closest* sink:

$$L_k = \min_{s \in S} L(k,s) \tag{1}$$

The maximal delivery latency in the network, L, characterizes the network performance.

$$L = \max_{k \in K} L_k \tag{2}$$

The optimization objective is to find, for a given number of sinks, the sink placement, i.e., the set of the nodes to act as sink nodes, such that L is minimized.

In its nature the SPO problem is multi-objective. The number of sink nodes in the network represents a trade-off between latency and deployment cost. For a given network topology, we explore this trade-off space by repeating an optimization procedure for different numbers of sink nodes.

IV. PERFORMANCE EVALUATION

A. Stochastic Variable Graph Model

The latency aspects of GWSN can be efficiently captured into a Stochastic-Variable Graph Model [4]. A stochastic-variable graph $G_{SV}(V,E,X)$, is a connected, directed graph. The vertices V represent nodes and the edges E represent direct links for data-item dissemination. The edge weights X are stochastic variables $X(i,j)$ with $(i,j)\in E$, which model the propagation delay from node i to node j. The stochastic variable integrates several stochastic models of different system aspects: communication (radio and MAC) and computation (gossiping and application). For example, the radio aspect is modeled with the probability that a transmission is successful, the MAC aspect with the probability that scheduling is successful (the receiver listens while the sender transmits), and gossiping with the probability that a data item is selected for transmission (see [4] for details). Stochastic models abstract from the protocol details, allowing short evaluation times, and providing the basis for fast and adequately accurate analysis.

B. Stochastic shortest path analysis

The SVGM models the propagation of a single data item independently of the propagation of other data items. The dissemination of a single data item is efficiently analyzed through a stochastic shortest path (SSP) calculation on the SVGM. Below, the main ideas of this approach [4] are briefly summarized.

Monte Carlo sampling of the SVGM provides concrete weighted graphs corresponding to the dissemination process of a specific data item. The edge weights of the graph, sampled from stochastic variables $X(i,j)$, are concrete propagation delays of the links. The dissemination along a single link starts as soon as the data item arrives at the source of the link and takes an amount of time according to the edge weight. Using the sampled graph, the time until the first arrival (latency) of a data item generated at node k to some other node s is calculated as the shortest path between nodes k and s in the sampled graph. This idea is illustrated in Figure 3, where the latencies of a data item generated at node 2 are analyzed. In this instance, the data item arrives at node 5, for example, after 6+5+7=18 time units.

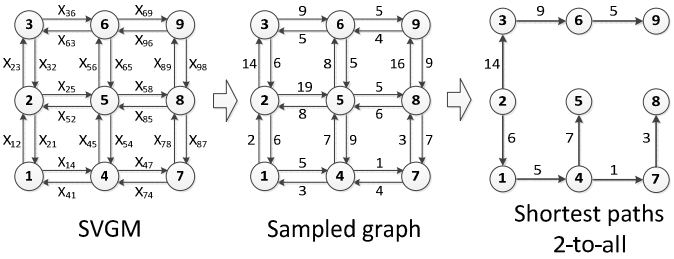

Figure 3. SSP calculation - Latency of a single data item (2) to all other nodes

In order to get statistically significant metric estimation, the propagation of sufficiently many data items has to be simulated by sampling new weighted graphs and calculating the shortest paths. The point-to-point latency is estimated as the average value over the simulated data items.

C. *All-to-sink* latency estimation

In order to estimate maximal delivery latency (Eqn. 2) the latencies from all source nodes to all sink nodes are required. Our SSP method (Figure 3) estimates the latency between two nodes as the average length of the stochastic shortest path connecting those nodes. The problem of all-to-sink latency estimation can be reduced to the standard (*one-to-all*) shortest path problem (Dijkstra [10]) by reversing the flow of data dissemination for the purpose of analysis, i.e., reversing directions of all edges in the sampled graph.

In the original SVGM method [4], for each data item a weighted graph is sampled from the SVGM and shortest path analysis is performed for each of the sampled graphs (Figure 3) to determine latencies. Reversing the shortest path analysis speeds up the performance evaluation process; instead of sampling a different graph for each data item source, we use a single graph to simultaneously sample latencies of data items from all sources. Note that in this case each of the sampled edge weights in the weighted graph represents at the same time the propagation delays corresponding to different data items towards the same sink. Although this sampling approach introduces artificial dependencies between dissemination of different data items, it does not affect properties of the point-to-sink metric estimation, i.e. the average length of the shortest path between two points is correctly estimated.

Both, reversed and direct [4] SSP methods simulate only a part of the system. The reversed calculates all-sources-to-single-sink latencies, while direct calculates single-source-to-all-sinks latencies. The evaluation time of the methods depends on the time required to sample random graph from the SVGM and the time to perform the shortest path analysis (using Dijkstra's algorithm). The SVGM sampling time (see [4] for details) is proportional to the number of edges in the graph, while Dijsktra's algorithm [10] has complexity of $O(|E|\log|V|)$, where $|E|$ denotes the total number of edges and $|V|$ the number of vertices in the sampled graph.

To calculate the latencies from all source nodes to all sink nodes, the reversed analysis has to be applied for each of the sink nodes. Straightforwardly using the direct method of [4], a separate evaluation would be required for each source node. As a result evaluation time of the reversed analysis is reduced proportionally to the ratio between the number of source nodes and the number of sink nodes.

V. SINK PLACEMENT

The partial performance evaluation method proposed in the previous section estimates *to-the-sink* latencies. In this section two variants of the greedy search heuristic driven with the results of the *to-the-sink* performance evaluation are proposed: a pure greedy and greedy simulated annealing.

A. The basic idea

The process of SPO is illustrated in Figure 4, where a small network with 21 nodes is given. The number of sinks to be deployed is two, and their positions have to be selected

among the existing 21 nodes. In each search iteration nodes are divided in two sets according to their closest sink, denoted with rectangles C_1 and C_2. Within each of the sets *the most critical node*, with largest latency, is identified (m_1 and m_2) Candidates for sink repositioning are the inbound neighbors of sinks (denoted with squares in the figure). The quality of repositioning candidates is assessed according to their contribution in the dissemination of data items from the most critical node to the sink. Each sink is then repositioned greedily to the best candidate position.

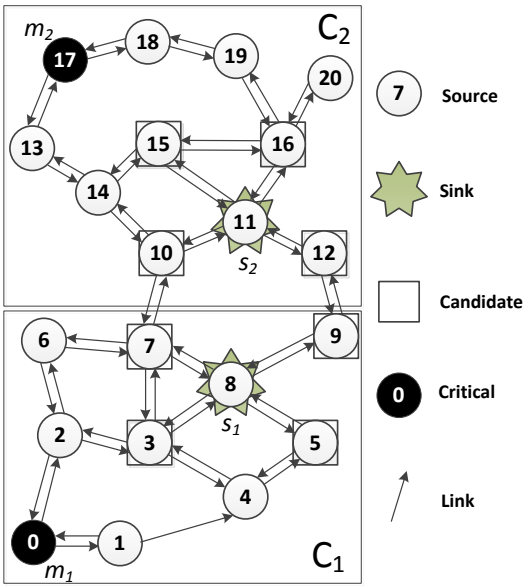

Figure 4. SPO algorithm

B. Details of the pure greedy search strategy

The pure greedy search is a simple optimization strategy, where in each iteration the algorithm makes the locally optimal (greedy) choice. The greedy search does not guarantee that the optimal solution is reached. However, in many cases it provides a near-optimal solution. The main advantage of a greedy search is its speed. The pure greedy algorithm for a given number of sinks is given in Figure 5. The optimization goal defined in Section III.C is to minimize the maximal average latency in the network. The greedy algorithm is performed iteratively, where in each iteration the algorithm repositions sink nodes such that the maximal average latency is maximally reduced.

The pure greedy search algorithm starts with an initial sink deployment. This initial sink deployment has an important effect on the search results. In Section V.C we address this issue. In each iteration the search algorithm performs the following steps:

1) *Estimate point-to-point latencies for each pair of source and sink nodes*. The network performance (Eqn. 2) is determined by the latencies to the closest sink. In each iteration, a new set of sink node candidates is computed as follows. The set of sink nodes selected in the *i-th* iteration is $S^{(i)}$. Then, for each sink $s \in S^{(i)}$, we create a list of source nodes for which sink s is the closest sink, C_s:

$$C_s = \{k \in V \mid \arg\min_{s' \in S^{(i)}} L(k, s') = s\} \quad (3)$$

2) *Identify performance critical aspects*. The network performance represents the maximal latency to the closest sink in the network. For a sink node s, the performance-wise most critical source node, m_s, is the source node from the set C_s with the maximal latency to that sink node:

$$m_s = \arg\max_{k \in C_s}(L(k, s)) \quad (4)$$

3) *Create the set of repositioning candidates for each sink node*. Our search strategy makes gradual changes in the sink placement, and allows that in a single iteration a sink node can be repositioned only to a position of one of its in-bound neighbors $IN(s)$, i.e., one of the nodes that can reach sink s at its current position:

$$IN(s) = \{i \in V \mid (i, s) \in E\} \quad (5)$$

The gradual, one-hop, change in considered sink positions, results in relatively small number of candidate locations and allows that the effect of the sink repositioning can be estimated efficiently as follows.

4) *Estimate the effect of sink repositioning*. The idea of our greedy search strategy is to move each sink node in a direction of its most critical source node and thus reduce the latency of data items originating at that source. We propose a simple heuristic that, based on the results of the performed partial analysis, estimates the benefit of the sink repositioning to specific nodes.

Our heuristic ranks the repositioning candidates based on their presence on the stochastic shortest paths. As repositioning candidates we use inbound neighbors of the sink node at its current position. A data item from the most critical node must arrive to the sink node via one of the repositioning candidates. For sink s and repositioning candidate $c \in IN(s)$, the *quality* of repositioning, $r_s(c)$, is computed as the fraction of data items originating at the most critical source node m_s that are delivered to sink s for the first time via repositioning candidate c. The fraction can be obtained from the stochastic shortest path analysis performed in step 1, as follows:

$$r_s(c) = \frac{\sigma_{ms}(c)}{N_{sim}}, \quad (6)$$

where N_{sim} denotes number of simulated data items per source node (i.e., the total number of random graphs sampled from SVGM, see Section III.B), and $\sigma_{ms}(c)$ is the number of graphs in which the shortest path from the most critical node m to sink s passes through the repositioning candidate c.

5) *Decide about new positions for all sink nodes*. For each sink node, a new position is chosen greedily, i.e., the best of the repositioning candidates is chosen according to the quality metrics calculated in step 4.

Algorithm 1 : Pure Greedy SPO
INPUT: $G_{SV}(V,E,X)$ - SVGM system model, n - number of sinks
1. Initial sink deployment: $S^{(0)}=\{s_1^{(0)},s_2^{(0)},...,s_n^{(0)}\}$
2. $i=0$;
3. **repeat** (until stop criterion is reached) - iterative optimization procedure
4. **for** *each* pair (k,s), $k\in V, s\in S^{(i)}$ calculate latencies:
5. $L^{(i)}(k,s)=perf\_eval(G_{SV},k,s)$
6. **end for**
7. **for** *each* $j\in\{1,...,n\}$ calculate:
8. Set of closest sources: $C_j=\{k\in V \mid \arg\min_s L(k,s)=s_j^{(i)}\}$
9. Most critical source: $m_j=\arg\max_{k\in C_s}(L(k,s_j^{(i)}))$
10. Set of repositioning candidates: $IN(j)=\{i\in V \mid (i,s_j^{(i)})\in E\}$
11. **for** *each* $c\in IN(j)$:
12. Estimate the quality $r_j(c)$;
13. **end for**
14. Choose the best candidate: $\hat{s}=\arg\max_c(r_j(c))$
15. $s_j^{(i+1)}=\hat{s}$
16. **end for**
17. $i=i+1$
18. **end repeat**

Figure 5. Pseudo-code for the pure greedy search strategy for SPO

C. Initial sink deployment

The greedy search strategy does not necessarily find a globally optimal solution, and often ends up in some local optimum. Our sink placement problem is discrete in nature. The greedy procedure is likely to guide search to a state where it will start oscillating between several positions. This is illustrated in Figure 6.

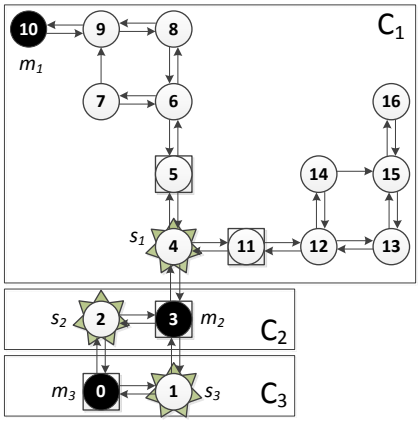

Figure 6. SPO algorithm stuck in a local optimum

The sink node s_1 starts oscillating between positions 4 and 5 when latency to the sink s_1 from node 10 is bigger than latency from node 16. The most critical node for the sink s_1 in position 4 is node m_1=10 and the best repositioning candidate is position 5. After sink repositioning (to the position 5) the most critical node becomes node 16 (instead of 10). Consequently, the new best candidate position will become the position of node 4, and the algorithm will start oscillating between positions 5 and 4.

The fact that nodes get stuck in a local optimum is not unexpected for a greedy search strategy. However, in combination with a bad initial deployment this issue can seriously diminish performance. Figure 6 gives an example. Assuming that links (4,5) and (4,11) are better than link (4,3), irrespective of the repositioning actions of sinks s_2 and s_3, sink s_1 remains in the oscillating state (notice that the content of the set C_1 does not change, and only stochastic effects in the latency estimations may accidentally lead the algorithm out of the oscillating state). The sink nodes s_2 and s_3 get trapped in a small part of the network by the sink s_1 that has reached a local optimum. The local optimum of the sink node s_1 prevents exploration for sink nodes s_2 and s_3. A different initial placement of sink nodes s_2 and s_3 might allow a better search scope, and potentially get node s_1 away from this local optimum.

A smart initial placement should try to ensure sufficient search scope for all sink nodes. Figure 7 proposes a simple *initialization* heuristic. Initially all sink nodes are placed in a single group in a network corner. One-by-one we reposition sinks to the position of the source with *the highest latency to the closest sink*. In that way sink nodes are spread over the network and *initial* interaction between their repositioning decisions is small. Applied to the scenario given in Figure 6, the initial sink deployment heuristic would (most likely, considering the stochastic nature of the algorithm) place sink nodes initially at positions 0,10, and 16.

Algorithm 2 : Initial sink deployment
INPUT: $G_{SV}(V,E,X)$ - SVGM system model, n - number of sinks
OUTPUT: $S^{(0)}=\{s_1^{(0)},s_2^{(0)},...,s_n^{(0)}\}$ - initial sink positions
1. $s_1^{(0)}=0$ - the first sink node is positioned to the location of node 0
2. **for** $j=1$ *to* $(n-1)$:
3. **for** *each* $k\in V$ calculate latencies:
4. $L(k,s_j^{(0)})=perf\_eval(G_{SV},k,s_j^{(0)})$
5. **end for**
6. Choose the position $\hat{k}$ with the *highest latency* to any of already positioned sinks $\{s_1^{(0)},...,s_j^{(0)}\}$: $\hat{k}=\arg\max_{k\in V}(\min_{s\in S} L(k,s))$,
7. $s_{j+1}^{(0)}=\hat{k}$
8. **end for**

Figure 7. Heuristic for initial sink placement

D. Greedy Simulated Annealing

In general, the local optimum found by a pure greedy search is determined with the starting search position. An alternative that provides still quick, but possibly improved solutions is a stochastic greedy search [18]. This approach, instead of selecting always the highest ranked repositioning candidate, performs weighted random selection of candidate positions. In this paper we incorporate the basic idea of the stochastic greedy heuristic in a simulated annealing search heuristic, and denote such approach as a greedy simulated annealing. The idea of simulated annealing is to start with a random search and decrease randomness as the procedure progresses. The process of the reducing randomness is referred to as *cooling* in the simulated annealing metaphor terminology. The randomness in the initial phase allows a wider search scope, which narrows down as the procedure progresses. The intention is to avoid low quality local

optima. The price that is paid for added randomness is initially slower convergence.

In our case, the randomness is added to the process of selecting the best repositioning candidate. In the first iteration the probability that a candidate position c is selected is proportional to its rank quality value. Then, the selection weights are linearly reduced over time, except for the weight corresponding to the best candidate, which is gradually increased. As a result after a predefined number of iterations (*cooling period* - T_c) the weight corresponding to the best candidate becomes dominant and the search behaves purely greedily. The selection weight of a repositioning candidate c in the i-th iteration of the search procedure is given as:

$$w(c) = \begin{cases} (1 - \frac{i}{T_c}) r_s(c) + \frac{i}{T_c} B(c), & i \le T_c \\ B(c), & i > T \end{cases}, \qquad (7)$$

where the *best variable B(c)* takes the value 1 in case that the candidate c is the best repositioning candidate; otherwise it is 0:

$$B(c) = \begin{cases} 1, & \arg\max_i (r_s(i)) = c \\ 0, & otherwise \end{cases}, \qquad (8)$$

The pseudo-code for the simulated annealing procedure is the same as the purely greedy procedure (Figure 5), with the exception of line 14. Line 14 should be replaced with the simulated annealing function (Figure 8).

Algorithm 3 : Simulated annealing - (replaces line 14 in Figure 5)
INPUT: s - sink, i - iteration number, n - number of sinks,
$IN(s)$ - set of repositinog candidates, r_s - quality estimations
OUTPUT: $\hat{s}$ - new sink position
PARAMETER: T_c - cooling period
1. Calculate temperature: $\alpha = i / T_c$ (defines randomness)
2. **for** *each* $c \in IN(s)$
3. **if** $(\arg\max_i (r_s(i)) = c)$: $B = 1$ **else**: $B = 0$
4. Assign selection weights: $w(c) = (1 - \alpha) r_s(c) + \alpha B$
5. **end for**
6. $\hat{s}$ = weighted_random_selection($IN(s), w(c)$)

Figure 8. Simulated annealing selection for new sink placement

VI. Experimental Evaluation

We performed an extensive set of simulations to analyze the proposed framework. We assume that each node in the network has an attached sensor and takes the role of data item source.

A. Simulation setup

Simulations are performed for topologies covering an area consisting of a few rectangular parts, similar to realistic scenarios, such as monitoring of buildings, green houses or out-door fields. Nodes are positioned in a grid-like structure with an average spacing R (Figure 9). Each node is allowed a random displacement from the grid position. The displacement is selected uniformly in the range (0, 0.4R) and independently for each axis. Network nodes are assumed to have a transmission range of $\sqrt{2}R$ (i.e., central nodes have 8 neighbors on average). A stochastic radio model is used to determine packet reception ratios, [8],[5]. The packet reception ratio (reception probability) for nodes whose distance is less than half of the transmission range is assumed to be one, and it linearly drops to zero for nodes at a distance equal to the transmission range.

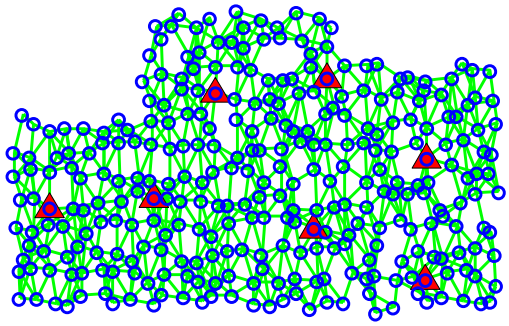

Figure 9. Grid-like topology and optimal sink deployment (sink placement is denoted by triangles)

The repositioning decisions are influenced by stochastic outcomes of the latency performance analyses. As a result, different repositioning decisions may be taken in two iterations where the sink nodes are in exactly the same locations. The statistical significance of the performance evaluation procedure and hence the variance in the outcomes depends on the number of simulated data items. The statistical significance is estimated using established techniques for estimation of long-run sample averages [7]. The performance evaluation procedure is run until the relative margins of error on the 95% confidence interval are below 2%. In order to achieve such statistical significance the number of simulated data items is set to be 3000 per source node. The chosen value for the relative margin of error is a compromise between evaluation speed and accuracy. All simulation experiments were run on a standard laptop with the CPU at 2GHz and with 4GB of RAM.

B. Performance of the SPO framework

The two main components of the proposed SPO framework are: partial performance evaluation and the search heuristic. In this subsection, first, we analyze the running time of the partial performance evaluation. Then, we compare the performance of different search strategies and discuss possible stopping criteria for the search heuristic.

1) Performance evaluation running time

The SPO framework iteratively searches for the best sink placement. The search procedure is guided with the result of the performance evaluation procedure. The performance evaluation takes a significant time even for a moderate margin of error of 2%. Our partial simulation method limits the simulation scope only to the processes relevant for the *to-the-sink* delivery. Such approach allows shorter evaluation times compared to the full system simulation as described in

[4]. The speed gain of the partial simulation approach in comparison with full simulation is presented in Table I. As expected, the speed gain is proportional to the ratio between the number of sources and the number of sinks.

TABLE I. RUNNING TIMES (IN SECONDS) AND ACHIEVED SPEEDUP, (ESTIMATIONS ARE MADE FOR 5 SINKS AND 3000 DATA ITEMS PER SOURCE).

Sim / Network size	**83**	**155**	**251**	**371**	**515**
Full sim. [s]	9.3	33.3	89.7	194.1	377.1
Partial sim.[s]	0.57	1.08	1.79	2.62	3.66
Speedup	16.6	31	50.2	74.2	103

The achieved analysis times allows embedding our framework in a full design flow where sink placement is only a part of large-scale design space exploration. For example, in the case when performance evaluation has to be performed 100 times (e.g., 100 search iterations), for a network consisting of 83 nodes and 5 sink nodes, the evaluation time is less than 1 minute (the number of simulated data items is 100*83*3000=24.9 million). For a network of 515 nodes, 100 iterations of performance evaluation take about 6 minutes (the number of simulated data items is 100*515*3000=154.5 million).

2) Comparison of different search methods

In this subsection we compare five optimization strategies. The first two optimization strategies considered are already described in Section V:

1. Pure Greedy (PG),
2. Greedy Simulated Annealing (GSA)

These strategies are compared with some other relevant strategies:

3. Random (R) – This approach does not apply any smart search heuristic; in each iteration, uniformly random sink selection is performed, independently from the selections made in previous iterations. The comparison with the random approach illustrates the improvements achieved by applying smart heuristics over purely random search.
4. Genetic (Gen) - The genetic strategy applies a genetic algorithm [29]. The genetic algorithm starts with a group of random solutions (the population) and in an iterative process modifies the population in a way that mimics the process of natural evolution and that improves the solutions in the population. It is a generally applicable optimization tool.
5. Approximate-greedy (AG) – The most time consuming procedure in the SPO framework is the performance evaluation. In order to get statistically significant results, we simulate 3000 data items per analysis. The approximate-greedy approach is a greedy heuristic that instead of using the stochastic shortest path analysis uses structural metrics to approximate the result. We use a simple shortest path analysis on a weighted graph, obtained by taking edge weights fixed according to the expected delays. Instead of performing the shortest path algorithm 3000 times, the shortest path analysis is then performed only once; however the stochastic aspect of the gossiping is not captured and performance is estimated less accurately.

The experiments are performed for networks consisting of 308 nodes and for different numbers of sinks $|S| \in \{5,7\}$. For each number of sinks, 30 random topologies are generated and results are presented as average values over all topologies.

To compare the various approaches, we consider the degree of optimality achieved versus the number of search iterations performed. The optimality (y-axis) denotes the ratio between the best obtained result for a given number of iterations (x-axis) and the optimal result found. There is no efficient way to obtain globally optimal solutions, and instead, as a reference, we use the best solution obtained by any of used optimization strategies in a complete search process (in this case 250 iterations).

Quality is assessed over a period of 250 search iterations. A search iteration evaluates the performance of a single sink placement solution. In the genetic algorithm, determining a new generation involves the modification of all population members. Thus, one generation is equal to several iterations. The time needed for one iteration corresponds to the time for one performance evaluation, and is thus the same for all search heuristics. Therefore, the horizontal axis in the graphs following can be interpreted as the computation time of the algorithm.

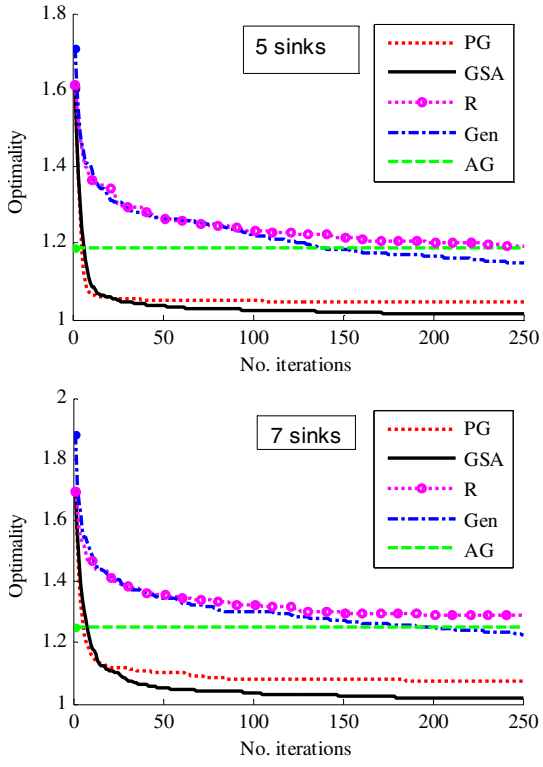

Figure 10. Comparison of five different optimization strategies in scenarios with a) 5, b) 7 sink nodes

The AG heuristic greedily searches a local optimum, using an extremely fast but possibly inaccurate approximation of latencies. Due to the extremely fast performance evaluation, AG search provides a result in a time shorter than a single iteration of any other method

(which all use the slower but more accurate SVGM-based performance evaluation). The notion of an iteration is therefore not meaningful for AG and the result is presented with a horizontal line in the figures below. An accurate performance evaluation is performed at the end of the AG search process in order to assess the quality of the end result.

Several interesting observations can be made from the results presented in Figure 10.

- The greedy strategy converges quickly to a good solution (20-30 iterations, depending on the scenario)
- Simulated annealing provides a better search scope than a pure greedy approach and therefore finds slightly better solutions (4% lower latency for 5 sinks and 6% for 7 sinks). As expected, simulated annealing initially converges a bit slower than a pure greedy approach, but the difference is negligible. The cooling period was set to 200 iterations.
- The random search converges slowly. In the scenario with 5 sinks, after 250 iterations the solution is 20% worse than the solution achieved with simulated annealing (the difference is 30% in case of 7 sinks).
- The convergence speed of the genetic approach in the initial phase is similar to the random approach, and improves over time. This result is expected. The random effects are dominant in the first generations of the genetic algorithm; as the evolutionary mechanism progresses, only the best individuals survive and the convergence speed improves. This result confirms our expectation that a genetic algorithm is too slow for large scale DSE problems with tight time budget.
- The number of sinks to be deployed affects the convergence speed of the search process. The more sinks, the slower convergence. This effect is more prominent in the genetic and random approaches, where the random effects are dominant, than in the greedy approaches where the search is guided.
- The approximate greedy approach provides solutions that are on average about 20% (25% for 7 sinks) worse than a solutions obtained by simulated annealing. However, such solutions are obtained very quickly and therefore can be useful in the initial phase of the GWSN design.

From the above observations, we can conclude that GSA provides the best compromise between convergence speed and quality of the end result. The performance of the solutions obtained after only 100 iterations is less than 2% off from the optimality reference. The generic and random search converge too slowly for use in DSE, the pure greedy approach has, as expected, a problem with avoiding bad local optima, while the approximate greedy is not accurate enough (although extremely fast).

3) Stopping criteria

The SPO methods iteratively reposition sink nodes, trying to improve overall performance. When the available time budget is very small, the convergence speed is of ultimate importance since the optimization procedure ends when the time limit is reached. When the time budget is more relaxed, ideally, the optimization procedure should be stopped at the moment when further (significant) improvements are not expected. We discuss two different criteria.

The greedy approach typically results in oscillations, e.g. the new iterations bring the search procedure to already explored solutions. The moment when the oscillations are detected can be used as a criterion to stop the search. This criterion cannot be used in a case of random procedures (including the stochastic greedy part of simulated annealing and the genetic algorithm) where nodes do not end in oscillations. In that case, so called stall time [29] can be used as the stopping criterion. The optimization procedure is stopped when no significant improvement is obtained for a period corresponding to the stall time. In the performed experiments a stall time of 40 iterations is sufficient.

C. *Trade-off analysis*

Finally, we use our SPO framework to explore the trade-off between numbers of sink nodes in a network (cost) and network performance. The experiments are performed on a set of random grid-like topologies with 251 nodes. The results presented in Figure 11 shows the reduction in latency achieved by increasing the number of sinks (normalized to the latency obtained for a single sink deployment).

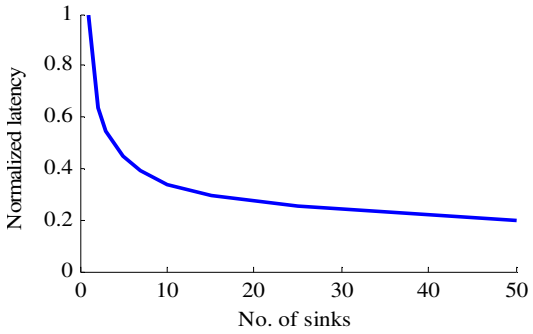

Figure 11. Latency vs. number of sinks

The more sink nodes are deployed, the smaller the delivery latency. The latency achieved with optimal deployment of 7 sink nodes is reduced to 40% of the latency achieved in case of single sink deployment on average. The latency decreases with the number of sinks, however the improvement achieved by adding new sinks to the network levels-off with the number of sinks. According to the presented results, we may conclude that in order to get good balance between cost and latency the number of sink nodes should be between 5 and 10 for typical networks in this set.

VII. CONCLUSION

The wireless sensor network designer requires a tool that can ease and accelerate deployment and provides efficient ways to improve operational efficiency. The sink placement optimization framework proposed in this paper can be useful in different phases of the design-time planning and dimensioning of gossip-based WSN. The combination of partial system analysis and a fast search strategy provides short running times and allows network designers to find adequate sink placements in acceptable time.

The proposed SPO framework finds a near-optimal placement that minimizes system latency, for a given number of sink nodes in a given network topology. The fast convergence of the greedy simulated annealing search makes this method the preferred solution in all scenarios with tight time budgets for sink placement analysis, e.g. in large-scale design-space exploration, where the sink placement procedure has to be performed many times.

Our case study limits candidate locations for sink placement to the locations where the sensor nodes are already placed. In case some other set of candidate locations is considered, the framework is still applicable, although the ranking metric for the repositioning candidates may have to be adjusted. Similar adjustments are necessary for different optimization criteria (e.g. average latency).

The statistical significance of the latency estimation has a profound impact on the quality of the deployment. Our results show that the difference between an approximate method that uses fast but statistically inaccurate estimation, and the more accurate SVGM-based stochastic analysis is significant (20-25% higher latency of the resulting solution).

The partial system analysis based on a Stochastic-Variable Graph Model (SVGM) has a wider scope than our SPO framework and can be applied in all situations when *to-the-sink* performance has to be estimated.

ACKNOWLEDGEMENT

This work was supported in part by the Dutch innovation program Point-One, through project ALwEN, grant PNE07007.

REFERENCES

[1] P.A. Anemaet, Distributed G-MAC: A flexible MAC protocol for servicing gossip algorithms. Master's thesis, Delft University of Technology, The Netherlands, 2008.

[2] R. Bakhshi, L. Cloth, W. Fokkink, B. Haverkort, Mean-Field Analysis for the Evaluation of Gossip. In Proc.QEST 2009, IEEE Computer Society pp. 247-256, 2009.

[3] R. Bakhshi, D. Gavidia, W. Fokkink, M. van Steen, A Modeling Framework for Gossip-based Information Spread In Proc. 8th Int'l Conf. on Quantitative Evaluation of SysTems (QEST 2011), Sept. 2011.

[4] M. Blagojević, M.Nabi, T. Hendriks, T. Basten, M.C.W. Geilen, Fast simulation methods to predict wireless sensor network performance. In Proc. PE-WASUN 2009, ACM, pp. 41-48, 2009.

[5] M. Blagojević, M.Nabi, M.C.W. Geilen, T. Basten, T. Hendriks, A Probabilistic Acknowledgment Mechanism for Wireless Sensor Networks. In Proc. Sixth International Conference on Networking, Architecture, and Storage, NAS 2011, ACM, pp. 63-72, 2011.

[6] A. Bogdanov, E. Maneva, S. Riesenfeld, Power-aware base station positioning for sensor networks, in Proc. 23rd Intl. Conf. INFOCOM'04, Hong Kong, 2004.

[7] J.Y. Le Boudec. Performance Evaluation Lecture Notes (Methods, Practice and Theory for the Performance Evaluation of Computer and Communication Systems). EPFL, Switzerland, 2010.

[8] A. Cerpa, J.L. Wong, L. Kuang, M. Potkonjak, D. Estrin, Statistical model of lossy links in wireless sensor networks. In Proc. of IPSN 2005, ACM/IEEE. 2005.

[9] P. Cheng, C.-N. Chuah, Xin Liu, Energy-aware node placement in wireless sensor networks, in: Proc. 47th IEEE Globecom'04, 2004.

[10] E.W. Dijkstra, A note on two problems in connexion with graphs. Numerische Mathematik, pp. 269–271,1959.

[11] E.A. Efrat, S. Har-Peled, J.S.B. Mitchell, Approximation algorithms for two optimal location problems in sensor networks. In Proc. 3rd Intl. Conf. on Broadband Communications, Networks and Systems (Broadnets'05), 2005.

[12] A. Fehnker, P.Gao, Formal Verification and Simulation for Performance Analysis for Probabilistic Broadcast Protocols. In ProcADHOC-NOW, pp.128-141,2006

[13] A. Guinard, A. McGibney, D. Pesch, A Wireless Sensor Network Design Tool to Support Building Energy Management. In Proc. 1st ACM Workshop On Embedded Sensing Systems For Energy-Efficiency In Buildings. 2009.

[14] T. Hérault, R. Lassaigne, F. Magniette and S. Peyronnet. Approximate Probabilistic Model Checking. In Proc. 5th Int. Conf. on Verification, Model Checking and Abstract Interpretation (VMCAI'04), Springer-Verlag, 2004.

[15] R. Hoes, T. Basten, C.K. Tham, M. Geilen, H. Corporaal, Quality-of-service trade-off analysis for wireless sensor networks, Performance Evaluation, Vol. 66, Issues 3-5, March 2009, pp.191-208

[16] S. Kim, S. Pakzad, D. Culler, J. Demmel, G. Fenves, S. Glaser, M. Turon, Health Monitoring of Civil Infrastructures Using Wireless Sensor Networks, In Proc. IPSN 07, Cambridge, MA, 2007.

[17] S. Kirkpatrick, C.D Gelatt, M.P. Vecchi, Optimization by Simulated Annealing, Science Vol.220 No.4598. pp. 671–680, 1983.

[18] V. Kodaganallur, A. Sen, Greedy by Chance – Stochastic Greedy Algorithms, In Proc. 6th Intl. Conf on Automatic and Autonomous Systems, 2010.

[19] A. Köpke, M. Swigulski, K. Wessel, D. Willkomm, P. Klein, T. Haneveld, T. Parker,, O. Visser, H. Lichte, S. Valentin, Simulating Wireless and Mobile Networks in OMNeT++ The MiXiM Vision. OMNeT++ Workshop, ICST, 2008.

[20] M. Kwiatkowska, G. Norman and D. Parker. Analysis of a Gossip Protocol in PRISM. ACM SIGMETRICS Performance Evaluation Review, 36(3), pages 17-22, 2008.

[21] H. Madhyastha, T. Anderson, A. Krishnamurthy, N. Spring, A. Venkataramani. A structural approach to latency prediction. In Proc. of Internet Measurement Conference, Rio de Janeiro, Brazil, 2006.

[22] W.Y. Poe, J.B. Schmitt. Self-Organized Sink Placement in Large-Scale Wireless Sensor Networks. In 17th MASCOTS 2009, pp. 629–631, 2009.

[23] K. Romer, Wireless sensor networks: From science to reality. In ¨ Proc. Intl. Conference on Sensor Technologies and Applications (SENSORCOMM'07), Valencia, Spain, October 2007.

[24] Y. Shi, Y.T.Hou, Optimal base Station Placement in Wireless Sensor Networks. ACM Trans. on Sensor Networks, Vol.5, No.4, 2009.

[25] Q. Wang, K. Xu, H. Hassanein, G. Takahara, Minimum cost guaranteed lifetime design for heterogeneous wireless sensor networks (WSNs). In Proc. 24th IEEE Intl. Performance, Computing, and Communications Conference (IPCCC'05), 2005.

[26] F. van der Wateren, The art of developing WSN applications with MyriaNed. Tech. report, Chess Company. the Netherlands, 2008.

[27] M. Younis, K. Akkaya, Strategies and Techniques for Node Placement in Wireless Sensor Networks: A Survey. J. Ad-Hoc Networks, vol. 6, no. 4, pp. 621-655, 2008.

[28] W. Youssef, M. Younis, Intelligent Gateways Placement for Reduced Data Latency in Wireless Sensor Networks. In Proc. Intl. Conf. on Communications (ICC'07), Glasgow, 2007.

[29] MATLAB R2012a Documentation - Global Optimization Toolbox, http://www.mathworks.nl/help/toolbox/

[30] http://www.isi.edu/nsnam/ns/

[31] www.omnetpp.org/

[32] www.opnet.com