On the Efficiency of Local Information-based Sink Deployment in Heterogeneous Environments

Zoltán Vincze
Budapest University of Technology and Economics
Department of Telecommunications and Media Informatics
vinczez@tmit.bme.hu

Rolland Vida
Budapest University of Technology and Economics
Department of Telecommunications and Media Informatics
vida@tmit.bme.hu

Attila Vidács
Budapest University of Technology and Economics
Department of Telecommunications and Media Informatics
vidacs@tmit.bme.hu

ABSTRACT

Deploying multiples sink nodes reduces communication distance, and thus energy consumption in wireless sensor networks. Determining exactly where to deploy those sinks so as to minimize the communication paths isn't however a trivial question. Solutions that use integer linear programming or iterative clustering techniques are usually not scalable, choose the sink positions only from a limited set of possibilities, and are based on global topology information. In this paper we present and analyze the *1_hop* algorithm, an approach that uses only local, last-hop routing information to iteratively find an efficient solution to the deployment problem. We show that the algorithm performs well in any kind of heterogeneous environment (heterogeneous sensor density, areas of irregular shape, or with obstacles), cases in which solutions that build on global topology information are often not appropriate.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Network topology

Keywords

Multiple Sinks, Sink Deployment, Wireless Sensor Networks

1. INTRODUCTION

Wireless sensor networks (WSNs) are in the focus of the research community both because of the large number of possible application scenarios, and the interesting open issues that arise from the specificities of the sensors and the wireless networks they can build. A sensor is a device that measures some environmental parameter (e.g., temperature, light conditions, radioactivity); such sensors are used in our everyday life since a long time, they are deployed in our cars, homes, and workplaces. Devices such as mercury thermometers, watt-hour meters, barometers, carbon monoxide detectors, or seismometers are well known and intensely used.

However, when talking about WSNs today, the research community has a different vision in mind. The technological progress made it possible to manufacture sensors at a microscopic scale, using e.g. the MEMS (Microelectromechanical Systems) technology. The goal is build sensors that provide significantly higher speeds and sensitivity, but on the other hand are cheap and tiny. Their size and low price permits then to deploy hundreds or thousands of them inside the area that has to be monitored. In addition, they are capable to communicate with each other through radio signals; thus, the measurement data can be transmitted, either directly or through intermediate relay modes, to one or several sinks. Nevertheless, these cheap and tiny devices have limited resources (memory, processing power, energy); therefore, special operation and communication schemes have to be designed and implemented in order to cope with these limitations.

One of the most important issues to handle is the energy consumption of wireless sensors. Similarly to the devices they power up, the sensor batteries should also be cheap and tiny. Moreover, it is usually impossible or impractical to replace or recharge them; therefore, sparing their energy is of paramount importance, so as to enable the sensors to perform their monitoring task as long as possible. Energy is consumed by several tasks, but most importantly by the communication module. If the covered area is small, sensors can send their data directly to the sink. In larger setups multi-hop communication is used, i.e., sensors forward each other's data toward the sink. In both cases the energy consumption depends on the communication distance between the sensors and the sink node(s).

We can reduce this communication distance in several ways. One solution would be to employ a mobile sink node, that can move either randomly [13][14], or following a predefined path [8][2], inside the covered region; the sink will collect the measurement data from a given sensor when the communication distance is assumed to be minimal (e.g., the sink moves inside the radio range of the sensor, or reaches the closest point on the predefined path). This solution isn't however acceptable for real-time applications, where data has to reach immediately the sink. In event-driven applications, when only those sensors talk to the sink that have

WICON 2007, October 22-24, Austin, USA

DOI 10.4108/pwsn.2007.2268

detected an unusual event, the sink can adaptively move closer to those active sensors [15]. However, if there are several simultaneous events that occur in opposite areas of the covered region, the sink cannot get close to one event if that would result in moving away from the other.

In this paper we will explore a different approach: reducing the communication distances by employing multiple sink nodes. This would decrease the average number of hops a message has to pass through before being received and processed by a sink, as data will always be sent to the closest sink. However, using multiple sinks is not enough by itself; the energy consumption depends largely on exactly where we deploy these sinks. This is a typical "facility location" problem: given a set of "facilities" (e.g., sink nodes) and a set of "costumers" (e.g., sensors) to be served from these facilities, where to deploy those facilities, and which facility should serve which costumer, so as to minimize the total "serving cost" (e.g., the overall energy consumption)?

In case of a single facility, the problem is already hard to solve. In the literature this problem is called the Fermat-Weber problem, and the point to find is the geometric median of the network, i.e., the point minimizing the sum of the distances between the sensors and the sink. This is of course equivalent with minimizing the average distance. Despite being an easy to understand concept, computing the geometric median is challenging. In [3] it was shown that there's no formula to solve it using only arithmetic operations and kth roots. There are however iterative solutions to approximate the geometric median, which produce a finer approximation after each step. One well known approach of this type is called Weiszfeld's algorithm [17], a form of iteratively re-weighted least squares (IRLS) solution. Its drawback is that it assumes global knowledge about the positions of the nodes. The problem is further complicated if several "facilities" (sink nodes) have to be placed. There are again solutions that address this issue based on integer linear programming (ILP) or iterative clustering techniques. However, these solutions are usually not scalable to large network sizes, they can choose only from a fixed set of possible locations, and most importantly, as the above mentioned Weiszfeld's algorithm, need a global knowledge of the network.

As opposed to these methods, in a previous work [16] we proposed the *1_hop* algorithm, a multiple sink placement solution that uses only local information: a sink has to be aware only of the positions of the sensors that are located inside its radio range, and of the number of remote sensors for which those neighboring nodes are last-hop relay nodes. In that previous paper we have shown that in the simplistic case of a square region, uniformly covered by sensor nodes and without any obstacles, the *1_hop* algorithm ensures comparable average communication distances to solutions that use global topology information to determine the optimal sink positions. Nevertheless, in realistic setups the monitored area has usually an irregular shape, coverage is not uniform, and there might be obstacles (e.g., walls, buildings) that obstruct the proper communication. These obstacles might even change their position in time, might appear or disappear at a given moment. To give an example, the empty shelves in a warehouse do not obstruct the communication between sensors, but if they are loaded, they might do so.

Therefore, in this paper we show that in such realistic cases the *1_hop* algorithm keeps performing very well, despite using only local information. By building only on the last hop routing information gathered at the sink, the algorithm does not have to be aware of any changes that might occur in the network size, the coverage density, or the obstacles that impede the communication inside the monitored region. As opposed to this, solutions that build on global topology information, an assumption already hard to defend, will be clearly outperformed in such a heterogeneous environment, unless they are kept aware of all the possible changes that influence the topology, a possibility that is usually unrealistic.

The rest of the paper is organized as follows. First, we present briefly the related work concerning sink deployment strategies. Then, in section 3 we describe two iterative solutions to approximate optimal sink positioning: the *global* algorithm uses global topology information, while the *1_hop* approach is based only on last hop, local routing information. In section 4 we describe the heterogeneous networking environment in which we would like to analyze the efficiency of these iterative solutions. Then, in section 5 we present simulation results for several realistic scenarios. Finally, the last section concludes the paper.

2. RELATED WORK

The deployment phase is a very important stage in the lifetime of a wireless sensor network. Most of the papers targeting deployment issues addressed only sensor node deployment, so as to ensure coverage and connectivity for the monitored area. However, sink node deployment is also crucial as far as energy consumption and network lifetime are concerned.

There are several papers that propose the use of multiple sinks to decrease the communication distances, and thus the energy consumption in the network. If there are multiple sink nodes, it is important to place them in an optimal way. A solution that we already mentioned in the previous section is the use of integer linear programming (ILP) [4][7][5][6][12][9]. This approach has however several drawbacks. On the one hand, it assumes a global knowledge of the topology; the optimal positions of the sink nodes are determined based on the locations of all the sensors, taking into account their energy levels, the costs of each communication link, etc. However, in a realistic setup, such knowledge cannot usually be assumed. On the other hand, the ILP solution can only find the optimal choice out of a limited set of possibilities. Also, it might be a time- and resource-consuming operation that is hard to repeat each time a topology change (e.g., depleted sensors, routing changes due to obstacles or load balancing) requires sink nodes to be repositioned. Moreover, the ILP approach does not scale for large networks with thousands of sensors.

A different approach for sink deployment is to use iterative clustering algorithms [10], such as $k-mean$. After defining some initial clusters, each sink node is placed in the center of mass of its own cluster. Then, the clusters are reshaped, as nodes are allowed to choose the cluster of the nearest sink (assuming the Euclidian distance as the clustering metric). This procedure is repeated iteratively until the clusters are not reshaped anymore. The main drawback of the approach is again the need for global knowledge. Moreover, being geographically close to a sink does not necessarily mean that there is a short routing path to that sink: obstacles or zones with depleted sensors might force the sensor to send its data to a sink that is farther away.

In [1] the sink deployment task is modeled as a maximum flow problem. First the authors show that finding out the optimum layout of the sink nodes is an NP-complete problem, even in the special case of homogeneous networks. Then, they analyze different heuristic algorithms that address the sink placement problem in the special case of a regular grid, but also for a uniform random graph or a preferential attachment graph. However, all these algorithms are centralized ones; the sink positions are determined based on global topology information. Moreover, the sink positions are chosen from a limited set of locations; sinks can be placed only at sensor locations.

Multiple sink nodes are also employed in applications such as target tracking, when several mobile sinks move inside the observed region, and all or some of them are interested in tracking a specific event. Nevertheless, this case is significantly different from the previous ones, as a sensor that detects the event has to alert all the interested sink nodes, instead of only the closest one. To provide this service, a multicast tree is built; the detecting sensor acts as a multicast source node, while the sink nodes act as multicast receivers [18].

In this paper we analyze a sink deployment strategy that is significantly different from all the above mentioned approaches. The sink nodes iteratively find their optimal positions, based only on local information; they need to be aware only of the positions of their neighboring sensors, and of the number of remote sensors for which these neighboring nodes are last hop relay nodes. Sink nodes can move anywhere inside the monitored region. Moreover, the network can be heterogeneous, can have an irregular shape, and the region can be filled with obstacles that obstruct communication. Sink nodes do not have to be alerted about any changes in the topology or the routing paths; they will change their positions, if necessary, based only on local information.

3. ITERATIVE ALGORITHMS TO APPROXIMATE OPTIMAL SINK POSITIONS

In [16] we gave a mathematical model for the optimal placement of one or more sink nodes in a multi-hop wireless sensor network, based on vector geometry. We considered that each sensor sends data to the closest sink. All sensors that send data to the same sink SK_i form a given cluster C_i. For each sensor ${s_j}^{(i)}$ of a specific cluster C_i, we defined an orientation vector ${\mathbf{v}_j}^{(i)}$ as the unit vector that points from the sink SK_i of that cluster towards that sensor node ${s_j}^{(i)}$. Thus, in each cluster C_i a resultant vector $\mathbf{R}_i$ can be calculated by adding up all the orientation vectors ${\mathbf{v}_j}^{(i)}$ starting from the corresponding sink SK_i. We showed that the average distance between the sensors and their closest sink is minimal if the resultant vectors inside each cluster are minimized. However, finding the specific positions where these resultant vectors are minimal is not trivial. Therefore, we proposed an iterative algorithm to approximate this optimal solution. We called this algorithm *global*, as it is based on global topology information.

3.1 The 'global' algorithm

We assumed that the sinks have global knowledge, i.e., every sink knows the geographical coordinates of the other sinks and of all the deployed sensors. Given an initial setup, each sink can decide which sensors are closest to him, and can divide the network into clusters, i.e., determine C_i, $i = 1 \ldots K$. Next, the algorithm uses an iterative procedure to determine the centroids of the clusters, i.e., the place inside each cluster where the resultant vector is minimal. In every cluster Ci the sink SK_i calculates first the resultant vector $\mathbf{R}_i$ corresponding to its current location. If $\mathbf{R}_i$ is above a certain threshold, then the sink moves in the direction determined by the resultant vector with a given step. This iteration is repeated until none of the sinks moves. As all the sinks have stopped, the clusters are recalculated ($C_i^{'}$) based on the new sink locations. If there is a cluster that changed, i.e., $C_i \neq C_i^{'}$ for some i then the procedure to determine the centroids of the clusters starts again.

3.2 The '1_hop' algorithm

In a realistic scenario the *global* algorithm is hardly applicable (if applicable at all). In a large sensor network with a huge number of nodes, deployed maybe randomly (e.g., thrown out of a plane over the covered region) it is not feasible for a sink to collect and store location information for every sensor.

Thus, in [16] we also proposed a different solution for the deployment of multiple sink nodes. We called this algorithm *1_hop*, as it uses only locally available information, gathered from nodes that are within 1 hop distance from a sink. The sink nodes know only the locations of the sensors that communicate directly with them ([11]) and the locations of the other sinks. In addition, they are also aware of the number of remote sensors these neighboring nodes are last hop relays for.

Given an initial setup, the sink nodes determine which neighboring sensors are communicating directly with them. Then, they wait for a certain time period, ensuring that within this interval every sensor sends at least one message to the sinks. The header of every message contains the ID of the original sender. When sink SK_i receives a message through a neighboring sensor ${s_j}^{(i)}$, it notes the ID of the original sender; thus, at the end of the period for each of its neighboring sensors the sink can determine the number of remote sensors for which they are last hop relay nodes. Based on this information, without any global knowledge on the topology, we can say that there are ${n_j}^{(i)}$ remote sensors "behind" a given neighboring sensor ${s_j}^{(i)}$. For each such remote sensor we define a unit vector ${\mathbf{v}_j}^{(i)}$ that points from sink SK_i towards the neighboring sensor ${s_j}^{(i)}$. One additional such unit vector is defined for the neighboring sensor ${s_j}^{(i)}$ itself. Then, for each sink we determine the resultant vector as the

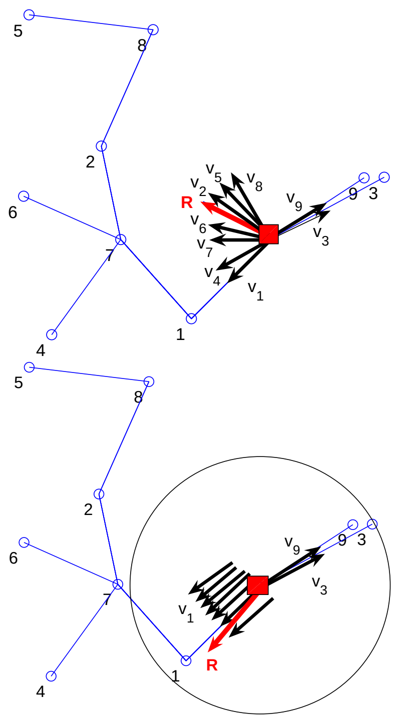

Figure 1: Computing the resultant vectors in case of the *global* (top) and the *1_hop* (bottom) algorithm.

sum of all unit vectors related to it. If this resultant vector is below a given threshold, the sink SK_i remains at its present position. Otherwise, it moves in the direction determined by the resultant vector, with a given step. As in the case of the *global* algorithm, this iteration is repeated until none of the sinks moves.

Note that during the iteration the resultant vector does not decrease in a monotone manner. Thus, the iteration is not able to reach the absolute minimum of the resultant vector, or at least the algorithm cannot decide at a certain step that it has reached the minimum, unless the resultant vector becomes 0. Nevertheless, if the vector size goes below a small enough threshold, we can say that we reached a sufficiently good position, so the algorithm can stop. Continuing the search for a better sink placement would result in unnecessary energy consumption (e.g., more location update messages should be broadcasted), which would probably not be compensated by the slight decrease in average communication distances.

Figure 1 illustrates the difference in the calculation of the resultant vectors between the two algorithms. The square denotes the sink, the small circles denote the sensors, while the lines represent the routes between the sensors and the sink. The big circle shows the radio communication range of a sensor, i.e. the sensors inside it are communicating directly with the sink. The unit vector pointing from the sink to sensor i is denoted by $\mathbf{v}_i$. When using the *global* algorithm the sink knows the location of every sensor; thus it can determine every unit vector, and then calculate $\mathbf{R}$ as the resultant vector of $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \ldots, \mathbf{v}_9$. As opposed to that, when using the *1_hop* algorithm the sink is only aware

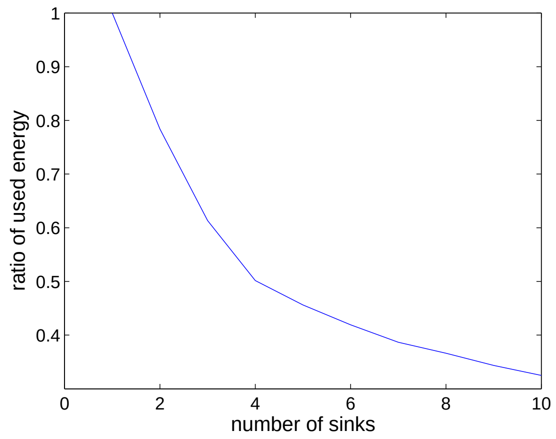

Figure 2: Energy consumption in function of number of sinks.

of the location of sensor nodes 1, 3 and 9. Then, the unit vectors corresponding to the distant nodes are replaced by the unit vectors corresponding to the last sensor node their packets travel through on the route to the sink. In this example this means that $\mathbf{v}_2, \mathbf{v}_4, \mathbf{v}_5, \mathbf{v}_6, \mathbf{v}_7$ and $\mathbf{v}_8$ are replaced by $\mathbf{v}_1$. Thus, $\mathbf{R}$ is calculated as the resultant vector of $\mathbf{v}_1, \mathbf{v}_1, \mathbf{v}_1, \mathbf{v}_1, \mathbf{v}_1, \mathbf{v}_1, \mathbf{v}_1, \mathbf{v}_3, \mathbf{v}_9$.

4. A HETEROGENEOUS NETWORK MODEL

The above presented *global* algorithm gives a good approximation of the optimal solution only in the case of a densely deployed, homogeneous sensor network, deployed over a region without any obstacles that might disturb routing. In such a network, packets sent by a given sensor are routed along the direct line that links that sensor to the closest sink. However, assuming such a network is unrealistic. There are many factors that can divert routing from that direct line. Sensor deployment might be heterogeneous, with sparsely populated areas where next hop nodes are chosen from a limited set of sensors, not necessarily located near that direct line. There might be obstacles in the area that have to be avoided. The monitored region might have an irregular shape, which could result in the direct line between a sensor and its sink crossing an area with no sensors deployed; in such a case the relaying along that line cannot be ensured.

In the realistic case of facing such a network model, heterogeneous from several points of view, the *1_hop* algorithm, based only on local routing information, is still very competitive, as we will show in the next section. All the above mentioned factors that divert routing from the direct lines will be implicitly noted and taken into account by the sink nodes when gathering information about their neighboring sensors and the paths that pass by them. The algorithm can be used in conjunction with any routing solution. For the sake of simplicity, in our simulations we assumed a shortest path routing given by the Dijkstra algorithm.

If sensors want to send data to the closest sink, they have to know its position. Relocating the sink nodes while searching

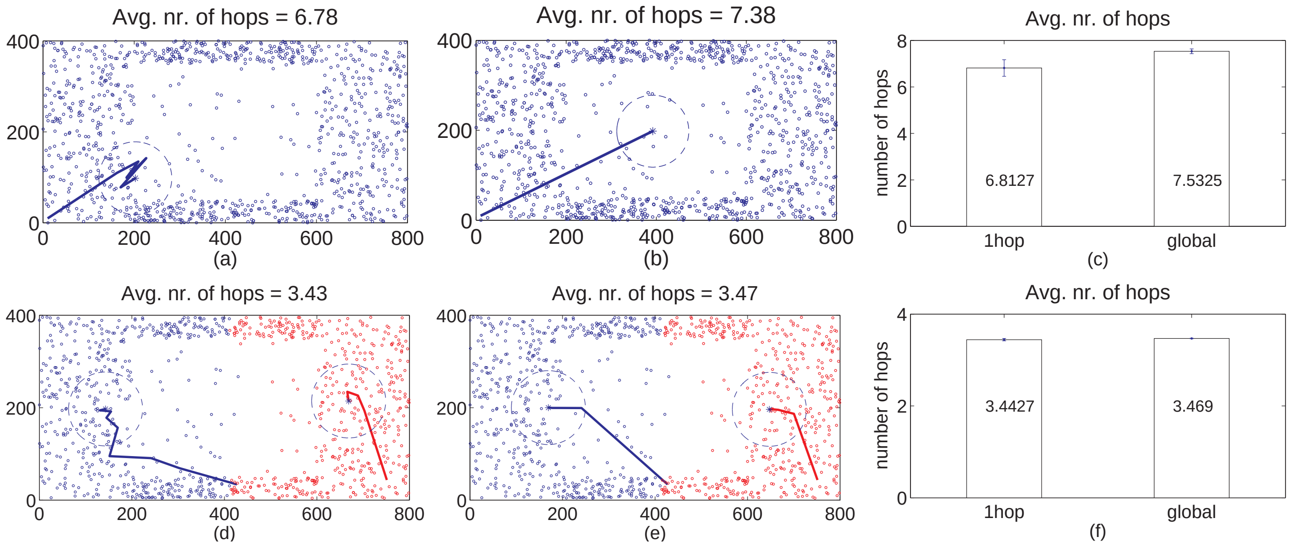

Figure 3: Sink placement in case of heterogeneous sensor density.

for their optimal position might have a negative side-effect as far as energy consumption is concerned, as sensors should be alerted about the changed sink positions through energy-consuming location update messages. However, there are several factors that can make this mechanism "power-friendly". Sink nodes might have the ability to cover the entire region through a single broadcast message, updating each sensor directly. Note that the sink nodes do not have power limitations; thus, they can afford such a "costly" update mechanism. Moreover, if needed, dedicated powered relay nodes can be deployed in the region to forward these update messages.

In [16] we have shown that in the simplistic case of a homogeneous sensor network, covering a regular shape area with no obstacles inside, the *1_hop* algorithm is quite as efficient, e.g., in terms of average communication distance, as the *global* algorithm, despite using only local information available at each sink. In the next section of this paper we will show through simulations, that in the case of a heterogeneous environment, as presented above, the *1_hop* algorithm clearly outperforms the *global* approach.

5. SIMULATION RESULTS

At the beginning of this paper we have argued that increasing the number of sinks will reduce communication distances, and thus energy consumption. However, the number of sink nodes has to be carefully chosen; too many sinks will result in high signaling traffic and high cost, while reducing only slightly the communication distances. Figure 2 addresses this issue. We have used a square shaped simulation area of 2200m x 2200m, over which we deployed uniformly but randomly 8000 sensors. The radio range of each sensor was 80 meters, and we used the *1_hop* algorithm to deploy the sink nodes. We continuously increased the number of deployed sink nodes, and we have measured the total energy consumption of the network, i.e., the total energy spent to send one packet from each sensor to the closest sink. We then divided this to the total energy consumption obtained when using only one sink node. For each scenario we had 5 simulation runs, the results presented here are the averages of these runs.

We can see that while increasing the number of sink nodes from 1 to 4, the energy consumption decreases sharply; when using 4 sinks we consume 50% less energy then in the case of only 1 sink. However, adding more sink nodes does not result in similar gains; adding 6 more sinks decrease energy consumption only with a further 18%. This depends of course on the density of the sink nodes and their radio range. If there is a large enough number of sinks, the communication distances are lowered to a small number of hops, and further reductions are hard to obtain.

The homogeneous deployment of sensor nodes, as assumed in the previous case and in most of the related papers, isn't however a realistic assumption. There are several reasons to consider a heterogeneous deployment: some areas may be less interesting or less important, some areas may be more hostile or hardly accessible, etc. Figure 3 shows that the *1_hop* algorithm performs very well in such situations. We have considered a rectangular region of 400m x 800m, over which we deployed 1050 sensors in the following way: in each of the two rectangles on the sides of the area we have put, randomly but uniformly, 300 sensors; in each of the narrow corridors, linking the two side rectangles on the top and the bottom of the area, we have deployed 200 sensors; finally, the middle part was populated with 50 sparsely deployed sensors. The radio range of each sensor was set to 60m.

In Figure 3(b) we can see that, in case of a single sink node, the *global* algorithm puts that sink in the middle of the area. However, due to heterogeneity of the network, and the sparsely populated middle area, that is not the optimal position. The *1_hop* solution, by taking into account only the last hop routing information, puts the sink into a position that ensures an average distance of 6.7 hops (Figure 3(a)), about 8% shorter than the 7.3 hops obtained for the

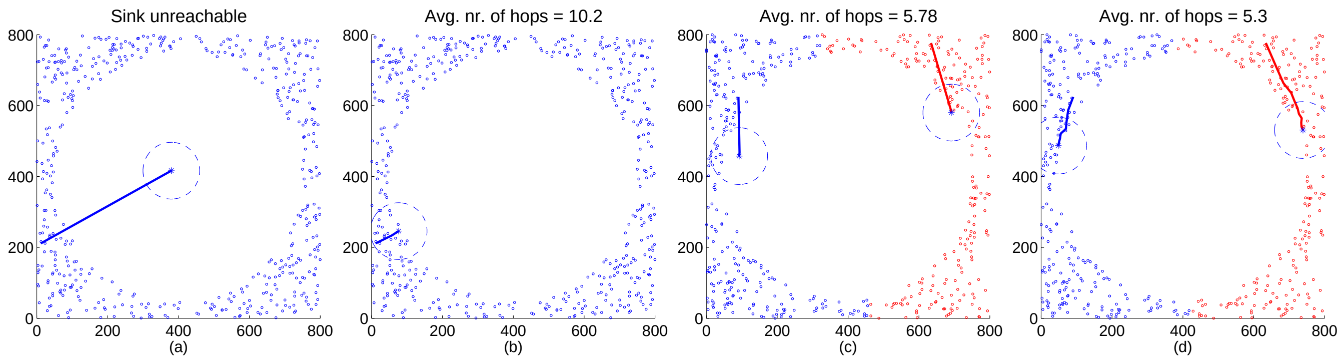

Figure 4: Sink deployment in areas of irregular shape.

global algorithm. Note that the thick blue line shows the movement of the sink node, from the original setup to its final position.

Moreover, the *global* algorithm puts the sink in the middle of the sparsely populated area; thus, the few sensors that can reach directly the sink (situated inside the circle that is drawn around the sink), will receive a high load to forward the packets of all the other sensors. They will therefore rapidly deplete their batteries, making the sink unreachable, unless relocating it. As opposed to this, the *1_hop* solution positions the sink in a denser area, with more nodes to handle the burden of last-hop forwarding. Consequently, it also spares the resources of the sparsely deployed nodes.

As we have previously said, the *1_hop* algorithm does not find the global optimum, it only finds a setup that ensures low-enough average distances, while using only local information. It is therefore clear that the quality of the solution might slightly vary in function of the setup the algorithm starts from. Figure 3(c) presents the average number of hops, as well as the 95% confidence intervals, obtained for the two algorithms in case of 10 different simulation runs. We can see that the efficiency of the *1_hop* algorithm depends more on the starting conditions. However, it ensures 10% shorter distances than the solution based on global information. When deploying one more sink, the efficiency of the *global* algorithm improves significantly (Figure 3(e)), as the sink nodes are placed outside the sparse area. Still, the *1_hop* algorithm performs slightly better, as shown both in the case of the same single initial setup (Figure 3(d)), and the case of 10 run averages (Figure 3(f)).

Most of the papers targeting wireless sensor networks assume monitored regions of regular shapes. However, this is far from being realistic. In the following we will show that the *1_hop* algorithm performs well for regions of any shape, as long as the network remains connected, and a routing algorithm ensures the communication between any sensor and the closest sink. For the sake of simplicity, let us consider a square area of 800m x 800m, with a circular shaped whole in the middle, having a radius of 350m (Figure 4). This can model for example a monitoring region with a lake, or a volcano crater in the middle. We deployed around the lake or the crater, i.e., inside the square but outside the circle, 500 sensors, uniformly but randomly. The radio range of the sensors was set to 80 meters. When using a single sink, the *global* algorithm places the sink in the middle of the circle (Figure 4(a)), this being the optimal position to minimize the Euclidian distances to the sensors. However, in that position the sink is not reachable by the sensors, does calculating the average distance does not make sense. Moreover, placing a sink in the middle of a lake or a crater does not make much sense either. As opposed to this, the *1_hop* algorithm places the sink outside the circle, and ensures reasonable communication distances (Figure 4(b)). When deploying two sink nodes, those nodes will be in a reachable position in both cases, although for the *global* algorithm the sinks will still be positioned inside the undesirable, empty region (lake or crater) (Figure 4(c)). As far as the communication distance are concerned, in this specific example the *1_hop* algorithm provides paths that are in average 9% shorter (Figure 4(d)). This example can be generalized to any area of any shape. In the realistic situation of a concave area, having a global knowledge about the positions of the sensors does not help us in avoiding to place the sink nodes in unreachable locations; to do that would require to have a global knowledge about the region boundaries and the sensor radio ranges as well. Besides being an overwhelmingly large amount of data to handle, these boundaries and radio ranges can even change in time in certain situations, which makes the solution even less appropriate. On the other hand, if a routing algorithm ensures the connection between sensors and sink nodes, the *1_hop* approach will determine valid and efficient sink positions for monitored areas of any shape, based only on local, last-hop routing information.

A special case of irregular areas is that of an area with obstacles that obstruct communication. Let us consider for example a square shaped area of 800m x 800m, with three straight walls of 600 meters each, deployed inside, as shown in (Figure 5). To build again a realistic analogy, we can see this scenario as the case of a warehouse, with walls or shelves that obstruct communication. We then deployed 800 sensors inside the entire area, with radio ranges set to 80 meters. We have also deployed 4 sink nodes, and used the *global* and the *1_hop* algorithms to position them. Then, we measured the average communication distances, in number of hops, between the sensors and their closest sink. The thick colored lines show the movement of the sink nodes in the different scenarios, while the coloring of the sensors shows to which sink they send the data.

As we have already mentioned, the efficiency of the deploy-

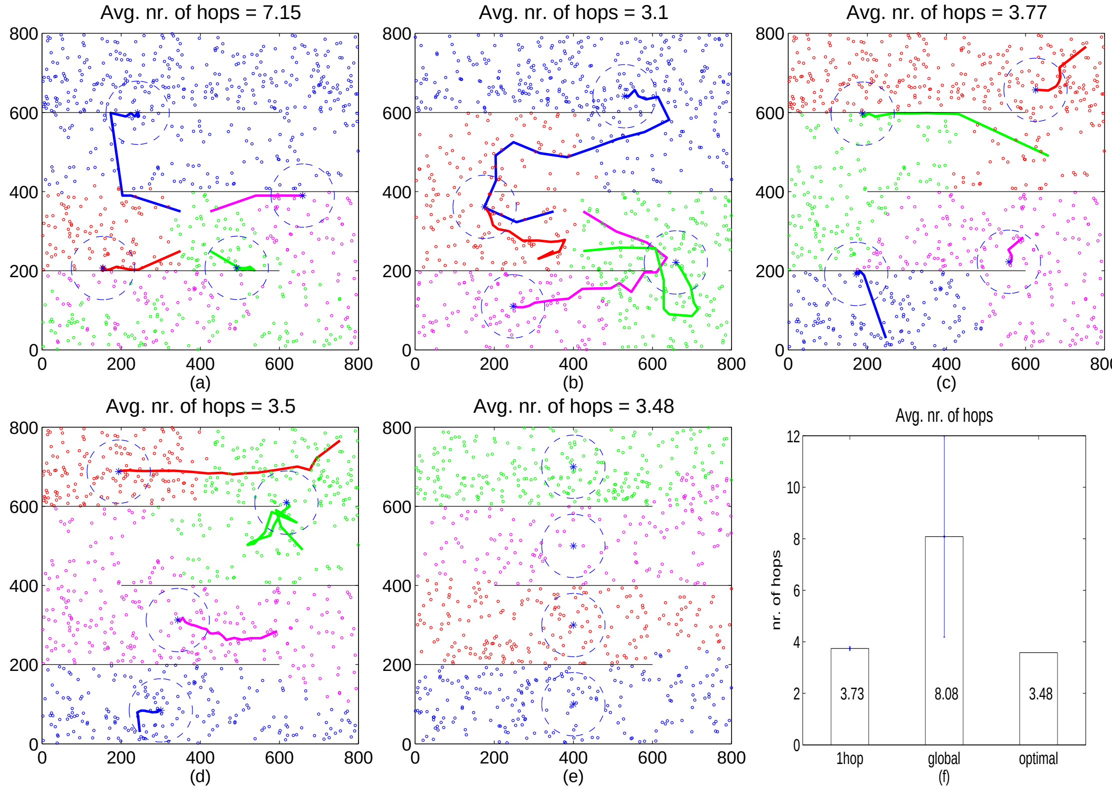

Figure 5: Sink deployment in area with obstacles.

ment solutions depends much on the original setup. If at the beginning the sink nodes are placed near to each other (Figure 5(a)), using the *global* algorithm does not allow them to spread much, resulting in high average distances. As opposed to this, when using the *1_hop* approach (Figure 5(b)), the sink nodes get positioned at quite distant points, reducing communication distances by more than 55%. Note also, that in the case of the *global* algorithm the sink nodes will often be placed just next to the walls; thus, several sensors in their immediate vicinity, but on the other side of the wall, will not belong to their cluster, as the paths that avoid those walls might be longer than the paths that lead to other, far away sinks.

When starting from random positions, the *global* algorithm might also result in short paths (Figure 5(c)), but note that even in this specific case, the *1_hop* approach gives slightly better results (Figure 5(d)).

In Figure 5(f) we show the average path lengths for 10 separate simulation runs, i.e., for the same topology we have chosen randomly 10 different starting points for the sink nodes, we have run the two algorithms for these setups, determined the average path lengths for each case, and calculated the average of these averages. We can see that the *1_hop* approach provides 54% shorter paths in average. Note also that while the deviation from that average in the case of the *1_hop* approach is extremely low, the efficiency of the *global* algorithm depends largely on the initial positions of the sinks.

Putting the sink nodes in the middle of the parallel corridors would seem to be a straightforward strategy to place the sinks in an optimal way (Figure 5(e)). As shown in Figure 5(f), the *1_hop* approach results in a nearly similar average distance as this "optimal" placement. However, note also that in the situation presented in Figure 5(b) the *1_hop* algorithm results in even shorter average paths than the supposed "optimum". This is due to the fact that there is a limited number of discrete sensors deployed over the area, and the random nature of the deployment might result in areas that are denser than others.

6. CONCLUSION

Most of the current research papers on sensor networks assume a homogeneous network model: uniformly deployed sensors over an area of a regular shape (square or circle), without any obstacles that might obstruct communication. They often also assume the knowledge of global topology information to perform centralized deployment or routing tasks. However, these assumption are usually not realistic.

In this paper we presented and analyzed the *1_hop* algorithm, an approach that uses only local, last-hop routing information to iteratively find an efficient solution to the deployment problem. We showed that this solution can be applied to any kind of heterogeneous environment; it performs well in case of varying sensor density, monitoring regions of irregular, even concave shapes, and inside areas that might obstruct communication.

7. REFERENCES

[1] A. Bogdanov, E. Maneva, and S. Riesenfeld. Power-aware base station positioning for sensor networks. In *Proc., IEEE INFOCOM 2004*, Hong Kong, Mar. 2004.

[2] A. Chakrabarti, A. Sabharwal, and B. Aazhang. Using predictable observer mobility for power efficient design of sensor networks. In *Proc., 2nd Int. Workshop on Information Processing in Sensor Networks (IPSN)*, pages 129–145, Palo Alto, CA, USA, Apr. 2003. Also in Lecture Notes in Computer Science, Vol. & (NO) (2634), pp. 129-145.

[3] R. Chandrasekaran and A. Tamir. Open questions concerning weiszfeld's algorithm for the fermat-weber location problem. *Mathematical Programming*, 44(A):293–295, 1989.

[4] S. R. Gandham, M. Dawande, R. Prakash, and S. Venkatesan. Energy efficient schemes for wireless sensor networks with multiple mobile base stations. In *Proc., IEEE GLOBECOM 2003*, volume 22, pages 377–381, San Francisco, CA, USA, Dec. 2003.

[5] Y. T. Hou, Y. Shi, H. D. Sherali, and S. F. Midkiff. Prolonging sensor network lifetime with energy provisioning and relay node placement. In *Proc. of Second Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks*, Santa Clara, Ca, USA, Sept. 2005.

[6] W. Hu, C. T. Chou, S. Jha, and N. Bulusu. Deploying long-lived and cost-effective hybrid sensor networks. In *Proc. of First Workshop on Broadband Advanced Sensor Networks*, Santa Diego, USA, 2004.

[7] H. Kim, Y. Seok, N. Choi, Y. Choi, and T. Kwon. Optimal multi-sink positioning and energy-efficient routing in wireless sensor networks. *Lecture Notes in Computer Science (LNCS)*, 3391:264–274.

[8] J. Luo and J.-P. Hubaux. Joint mobility and routing for lifetime elongation in wireless sensor networks. In *Proc., IEEE INFOCOM 2005*, Miami, FL, USA, Mar. 2005.

[9] P. Maulin, R. Chandrasekaran, and S. Venkatesan. Energy efficient sensor, relay and base station placements for coverage, connectivity and routing. In *Proc. of 24th IEEE International Performance, Computing, and Communications Conference(IPCCC 2005.)*, Phoenix, USA.

[10] E. I. Oyman and C. Ersoy. Multiple sink network design problem in large scale wireless sensor networks. In *Proc. of ICC 2004*, Paris, France, 2004.

[11] R. Peng and M. L. Sichitiu. Angle of arrival localization for wireless sensor networks. In *Proc. of the Third Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks*, Reston, Va, USA, 2006.

[12] L. Qiu, R. Chandra, K. Jain, and M. Mahdian. Optimizing the placement of integration points in multi-hop wireless networks. In *Proc. of International Conference on Network Protocols(ICNP)*, Berlin, Germany, 2004.

[13] S. J. Rahul C. Shah, Sumit Roy and W. Brunette. Data MULEs: Modeling a three-tier architecture for sparse sensor networks. In *Proc., IEEE Workshop on Sensor Network Protocols and Applications (SNPA)*, pages 30–41, Anchorage, alaska, USA, May 2003.

[14] L. Tong, Q. Zhao, and S. Adireddy. Sensor networks with mobile agents. In *Proc., IEEE MILCOM 2003*, volume 22, pages 688–693, Boston, MA, USA, Oct. 2003.

[15] Z. Vincze, D. Vass, R. Vida, A. Vidács, and A. Telcs. Adaptive sink mobility in event-driven densely deployed wirelss sensor networks. *Ad Hoc and Sensor Wireless Networks*, 3/2-3(A):255–284, 2007.

[16] Z. Vincze, R. Vida, and A. Vidács. Deploying multiple sinks in multi-hop wireless sensor networks. In *Proceedings of ICPS 2007, IEEE International Conference on Pervasive Services*, Istanbul, Turkey, 2007.

[17] E. Weiszfeld. Sur le point pour lequel la somme des distances de n points donnes est minimum. *Tohoku Mathematical Journal*, 43:355–386, 1937.

[18] W. Zhang, G. Cao, and T. L. Porta. Dynamic proxy tree-based data dissemination schemes for wireless sensor networks. In *IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, Fort Lauderdale, FL, USA, Oct. 2004.