Redeployment of Randomly Deployed Wireless Mobile Sensor Nodes

Khalil Mougou*†, Saoucene Mahfoudh‡, Pascale Minet† and Anis Laouiti‡
*ENSI, Campus universitaire La Manouba, 2010, Tunis, Tunisia,
Email: khalil.mougou@gmail.com
†INRIA, Rocquencourt, 78153 Le Chesnay Cedex, France,
Email: pascale.minet@inria.fr
‡Telecom SudParis, 9 Rue Charles Fourier, 91000 Evry, France,
Email: saoucene.ridene@it-sudparis.eu, anis.laouiti@it-sudparis.eu

***Abstract*—Wireless sensor networks (WSN) are generally randomly deployed in a given area. This initial deployment does not achieve neither area coverage, nor network connectivity. Thus, a redeployment algorithm has to be applied in order to achieve these two goals. This algorithm should meet performance criteria like saving energy and achieving stability. Our contribution in this paper is the design of DVFA, a distributed redeployment algorithm based on virtual forces. We simulate its behavior with NS2 and compare its performances with CVFA, a centralized version also based on virtual forces.**

I. Context and motivations

A sensor is a device able to monitor its environment by collecting information. A wireless sensor network, in short WSN, is a collaboration between these devices that are deployed in a given area and communicate by means of radio. The spectrum of applications supported by WSNs is very huge, we may have: environmental applications, intrusion detection, survivors detection, health care applications, ambient assisted living, industrial process control and military applications.

In this paper, we focus on applications where any event occurring in the monitored area must be reported either to a sink in the centralized case or to a group of specific nodes in the distributed case. Hence, all the considered area must be covered by sensors ensuring that any event is detected in the sensing range of at least one sensor. In addition, the sensor network must be connected in terms of radio communication in order to forward the detected event to the sink(s). Furthermore, we assume that each sensor is able to move and has a limited amount of energy. The initial deployments of these sensor networks are generally random and do not ensure the coverage of the given area. Moreover, battery depletion can also induce coverage hole or connectivity loss. Some zones may have a high coverage degree whereas others are poorly covered. Thus, a redeployment algorithm has to be applied in these cases. That is why, we propose a distributed redeployment algorithm whose aim is to redeploy the sensor network to ensure coverage and connectivity, while minimizing the distance traveled by each sensor node.

This paper is organized as follows: In section II, we deal with the state of the art. Section III presents the centralized virtual forces algorithm called CVFA. In section IV, we describe the principles of DVFA, our distributed algorithm based on virtual forces. In section V, we evaluate the performances of our algorithm and compare them with those of the centralized version of the virtual forces algorithm. We then discuss the stability issue and give some potential solutions in Section VI. Finally, we conclude in Section VII.

II. State of the art

Assuming an initial random deployment with possibly coverage holes and redundant sensors in some zones, how to redeploy theses nodes in order to achieve the desired coverage degree and network connectivity? Existing redeployment algorithms can be classified into centralized and distributed algorithms. The first designed algorithms were centralized. Some are too complex or consume too much resources to be implemented by the sensor nodes. Consequently, they only exist in a centralized version. Some of them were extended to work in a distributed way. In the following, we overview some existing centralized and distributed redeployment algorithms.

A. Centralized Algorithms

The first redeployment algorithms were centralized. A central entity knows the information, like position and energy, of all sensors, computes the new virtual positions of sensors running a given algorithm. At the end of the algorithm, it communicates the final positions to sensor nodes that move towards them. We list here three main examples:

- **Particle swarm optimization**, denoted PSO. In a particle swarm, each particle is assumed to know its best position reached during the algorithm execution, position denoted $pbest$, as well as the best global position reached by the swarm, position denoted $gbest$. Each particle computes an elementary move taking into account $pbest$ and $gbest$. This move is made only if the computed position is better than the current one. See [2] and [3] for more details.
- **Virtual forces**, denoted VF. In [6] and [7], each sensor node exerts a force on each neighboring sensor node. This force is attractive, repulsive or null depending on the distance between them. The goal is to reach the target distance, where no force is exerted. The new location of the sensor node is computed from the resulting

978-1-4673-1881-5/12/$31.00 ©2012 IEEE

forces. The computations are done by a central entity and sensor moves are virtual as long as the algorithm is not ended. The algorithm halts when the maximum number of iterations is reached or the equilibrium is obtained. Only at the end of the algorithm, each sensor moves to its final position, previous moves were virtual.

- **Hybrid algorithms** combining both VF and PSO. In [4], authors consider both static sensor nodes and mobile ones. Virtual forces are used to improve the convergence of PSO.

Other algorithms to redeploy sensors are given in [5]. Authors show how cascading moves to replace a failed sensor contribute to increase network lifetime by a more efficient use of energy: several sensor nodes make a small move instead of a single sensor node making a long move and then consuming a large amount of energy.

B. Distributed Algorithms

Many papers have proposed distributed algorithms to achieve maximum coverage of an area. The main target is to find a uniform deployment of the network in a minimum time, having in mind the energy constraints of such autonomous devices. Nodes will rely on their local neighboring knowldege to take the decision to move in order to reach a stable position. All of them use in some way the concept of forces like in the model of virtual forces that we have previously seen.

- **Distributed Self-Spreading Algorithm** (DSSA) [11] is an algorithm inspired by the equilibrium of molecules. In DSSA, each node calculates, at each step, the total forces exerted on it by its direct neighbors (located within communication range) in order to decide its movement. A node will stop its movement if it travels a distance less than a predefined distance e within a time duration S_{lim}. All the nodes seem to move together on each step, which is not easy to achieve in a distributed manner if the nodes are not synchronized.
- **A Force-based Genetic Algorithm** [10] is a decentralized topology control mechanism running on several nodes to achieve a uniform spread and hence accomplish the coverage of an area. A mobile node uses the sum of the forces exerted by the neighbors to decide which direction to take. This sum is actually the fitness function. A small value of the fitness funtion corresponds to a situation with a weak total force exerted on a node which means a more stable and better position. Each node running FGA generates r chromosomes corresponding to r possible positions and representing the possible solutions for the next generation. A node runs the FGA algorithm for g generations and selects the chromosome with the best fitness value, which means selecting the most suitable direction to take. The major drawback of the genetic algorithm is the need of computational power in order to explore the numerous possible choices.
- **the Mass-Spring-Relaxation algorithm** [8] and [9] are inspired from the mechanic physics laws. They work in a fully distributed manner based on the local knowledge of the neighbors. They consider that no global positioning system is available, nodes rely on their ultrasound sensors to detect their immediate neighbors and estimate their distances. The lack of global positioning system increases the overall complexity of the algorithm in order to resolve the misplaced node cases and avoid the node jumps.

Solutions based on Virtual Forces Algorithm, VFA, have a faster convergence, as observed in [3], and are less complex than particle swarm optimization for instance. That is why, we chose this algorithm and decided to design a distributed version. First, we detail its behavior.

III. Centralized Virtual Forces Algorithm

In this section, we start by describing the virtual forces algorithm principles followed by the enhancement that we have proposed in [1]. Basically, the algorithm is a loop where the stop condition is given by the full coverage or the maximum number of iterations reached. At each iteration, the algorithm computes for any sensor s_i located at (x_i, y_i), the force, denoted $\overrightarrow{F_{ij}}$, exerted by any sensor s_j located at (x_j, y_j). Let d_{ij} be the euclidian distance between s_i and s_j and D_{th} the target distance (i.e. ideal distance between two neighboring sensors). The force $\overrightarrow{F_{ij}}$ is:

- Attractive if $d_{ij} < D_{th}$. We have $\overrightarrow{F_{ij}} = K_a(d_{ij} - D_{th})\frac{(x_j - x_i, y_j - y_i)}{d_{ij}}$, where K_a is a positive coefficient;
- Repulsive if $d_{ij} > D_{th}$. We have $\overrightarrow{F_{ij}} = K_r(D_{th} - d_{ij})\frac{(x_j - x_i, y_j - y_i)}{d_{ij}}$, where K_r is a positive coefficient;
- Null if $d_{ij} = D_{th}$.

The resulting force exerted on s_i is equal to $\overrightarrow{F_i} = \sum_j \overrightarrow{F_{ij}}$. The new location of sensor s_i is given by (x_i, y_i) with $x'_i = x_i + x\text{-}coordinate\ of\ \overrightarrow{F_i}$ and $y'_i = y_i + y\text{-}coordinate\ of\ \overrightarrow{F_i}$. When the new positions of all sensor nodes have been computed, they become the current positions for the next iteration. However, we can notice some limitations of VFA.

Forces exerted on a sensor computed at iteration k take into account the sensor locations computed at iteration $k-1$. Since these nodes move at iteration k, the computation of the resulting force does not take into account the new location of these sensors. Furthermore, VFA does not take into account the existence of sensors with low residual energy, all sensors have to move according to the VF principle. We also observe that sensors move on large distances, making the redeployment energy greedy. A second observation is the existence of some zones that are better covered than others. In a weakly covered zone, the failure of a sensor can lead to loss of coverage and connectivity. That is why we propose CVFA in [1] to improve:

- *Coverage and network connectivity* by (i) taking into account at iteration k the new location of sensors in this iteration for the computation of the forces exerted on a sensor and (ii) serializing sensors movements according to a priority;

- *Energy efficiency* by (i) defining sensor node priority as a function of the residual energy of the sensor and (ii) limiting the maximum distance traveled by any sensor in an iteration;
- *Robustness degree* by uniformly deploying the sensor nodes in the monitored area. This will lead to a uniform robustness degree.

CVFA will be used as a reference in performance evaluation of the distributed version we propose in this paper. We now describe our distributed version of virtual forces algorithm, called DVFA.

IV. Principles of DVFA

In this section, we present DVFA, Distributed Virtual Forces Algorithm, where each sensor node is autonomous and periodically sends its $Hello$ messages.

A. Assumptions

In this paper, we adopt the following assumptions:

- Each sensor node is able to determine its own position,
- Each sensor node has a processing and a moving capacities,
- Each sensor node has a detection range r and a communication range R,
- Each sensor node knows D_{th} the target distance,
- The number of sensors, N, is sufficient to cover the target field.

In this paper, we assume that $R \geq 2r$. In such conditions, full coverage implies network connectivity.

B. General principles

We now describe the behavior of our algorithm in charge of redeploying wireless sensors to ensure both area coverage and network connectivity. Since the initial deployment is random, it may produce connectivity islands or connected components. Our algorithm, called DVFA, will progressively fusion them to reach a uniform coverage. Sensor moves are determined by the Virtual Forces principle. DVFA is highly adaptive to any environment. In fact, it naturally adjusts its parameters according to the new discovered connected components to achieve its goals. Moreover, arrivals and departures of nodes are seamlessly taken into account without resorting to a central entity or specific processing. DVFA works as follows. Each sensor node detects the arrival of new neighbors and the departure of others by means of $Hello$ messages exchanged periodically. The $Hello$ message contains the position of the sender node and the list of its 1-hop neighbors. The collected information allows each node to compute the forces exerted on it by nodes at a distance less than or equal to D_{Limit}, where D_{Limit} is the maximum distance within which virtual forces are exerted (at most 2-hop neighbors), as explained in Section III. After the computation and before moving, any node sends a Bye message including its new position after its next move. This message enables the neighboring nodes to update immediately their neighborhood tables and compute a more accurate value of the virtual forces exerted on them. Finally, the node moves to its new position. Notice however, that a sensor node is never allowed to move more than $Lmax$ in a single time period.

The timing behavior of any sensor node in DVFA is organized as depicted in Figure 1. Notice that there is no synchronization between sensor nodes. Each sensor follows its timing scheme independently, comprising three durations: T_{Hello} the $Hello$ period, T_{Lmax} the time needed to travel the maximum distance allowed and T_{FV} the maximum time needed to compute the virtual forces. We have $T_{Hello} = \varepsilon + T + T_{FV} + T_{Lmax}$, where ε is a random time selected in $[0, \varepsilon_{max}]$ to avoid the contention between all nodes arriving simultaneously at their new position. After a delay $T + \varepsilon$, it computes the virtual forces exerted on it, taking into account both the $Hello$ and Bye messages received to compute its new neighborhood. Periodically, each sensor node sends a $Hello$ message. Finally, it sends a Bye message before moving to its new position that it reaches after a delay at most equal to T_{Lmax}.

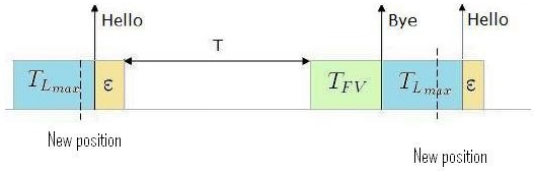

Fig. 1. Timing scheme in DVFA

V. Performance evaluation

We have implemented the DVFA protocol as an agent in the NS2 simulator and have performed simulations for different wireless sensor networks. Simulation parameters are given in Table I.

TABLE I
Simulation parameters

Topology	
Sensor nodes	170, 210, 250...330 randomly distributed
Area size	500m x 500m
Speed	2m/s
Simulation	
Result	average of 10 simulation runs
Max. number of iterations	1000 for CVFA
Simulation time	500s for DVFA
MAC	
Protocol	IEEE 802.11b
Throughput	2 Mb/s
Radio range R	50 m
Sensing range r	25 m
DVFA	
Ka	0.001
Kr	0.5
Hello period	2.5 s
$Lmax$	$D_{th}/6$
D_{Limit}	$2R$

A. Evaluation criteria

The goal of DVFA is to obtain the maximum coverage with the redeployment of network nodes. To compute the coverage rate, we virtually divide the network area into 500x500 unit grids. A grid is considered covered if and only if its centered point is covered by at least one sensor node. In this performance study, we will evaluate: the coverage rate, the total distance traveled and the time to obtain the maximum coverage rate. We compare the performance results of DVFA with the centralized algorithm CVFA.

B. Impact of the number of sensors on the coverage

Figure 2 illustrates the coverage rate obtained with DVFA and compares it with the one given by CVFA. We remark that this coverage is higher than 98% for all simulation scenarios using DVFA. Moreover, DVFA and CVFA have very close and very good performances which reach 100% with 330 nodes.

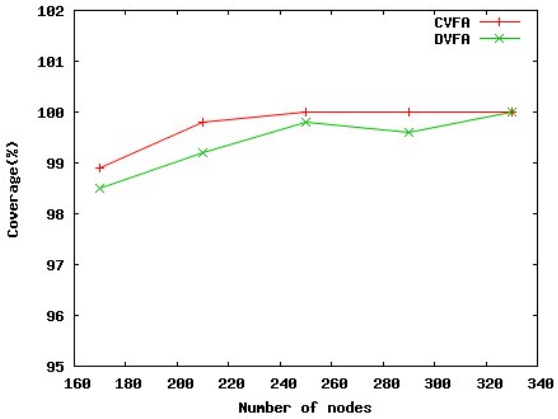

Fig. 2. Impact of the number of sensors on the coverage

In Figure 4, we present the time taken to reach the maximum coverage with DVFA. In case of CVFA (Figure 3), we present the number of iterations used by CVFA to reach the maximum coverage. This number gives an idea about the complexity of this algorithm. We notice, that with CVFA, the number of iterations is very high and reaches the maximum in case of 170 nodes, without ensuring a 100% coverage. This number decreases significantly when the number of nodes increases: a higher number of nodes contributes to a faster convergence. Notice that the main difficulty in CVFA is to collect the initial node positions and disseminate the new computed ones, because of possible disconnected communication islands.

However, with DVFA, the time taken to reach the maximum coverage is difficult to evaluate because of a potential unstability. In fact, nodes continue computing their virtual forces and so keep moving. Therefore, the maximum coverage reached can be lost. A good property of DVFA is that the coverage remains between 98% and the maximum reachable coverage. Nodes movement (or oscillations) after reaching maximum coverage is useless for protocol performances and causes nodes energy loss. This oscillation issue will be discussed in section VI.

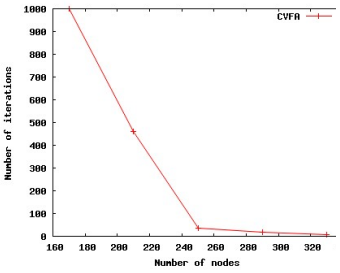

Fig. 3. Number of iterations to reach maximum coverage with CVFA.

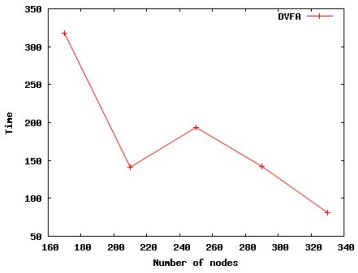

Fig. 4. Time to reach maximum coverage with DVFA.

Figures 5 and 6 show nodes distribution with CVFA and DVFA respectively. The number of nodes in this case is 210. We can remark that node density is almost the same for CVFA and DVFA. CVFA reaches a perfectly regular distribution respecting the hexagonal rules (each node, except nodes in the border, has six neighbors).

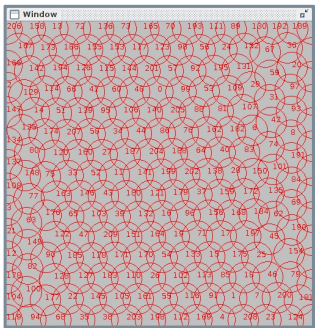

Fig. 5. Nodes distribution with CVFA.

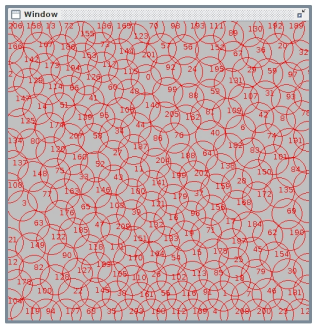

Fig. 6. Nodes distribution with DVFA.

C. Distance traveled by sensors

Figure 7 illustrates the total distance traveled by nodes to obtain maximum coverage. As expected, the total distance traveled by nodes using DVFA is much higher than the one obtained with CVFA.

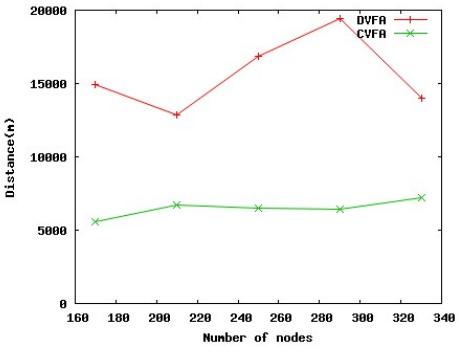

Fig. 7. Total distance traveled by sensors to reach maximum coverage

In fact, with DVFA, each node computes its new position based on information collected from neighbors. Then, it

moves to this new location and recomputes its next position and so on. However, in the case of CVFA, only the central node executes the CVFA algorithm until obtaining maximum coverage rate. Then, it disseminates the new nodes locations. Thus, sensor nodes only move to their new and final positions.

Figures 8 and 9 show the additional cost in time and traveled distance to enhance the coverage from 98% to the maximum reachable coverage. This enhancement of at most 2% of the coverage would come with a significant additional cost (e.g. tripled traveled distance and time). A coverage of 98% is sufficient for some applications like intrusion detection, since all paths can be detected.

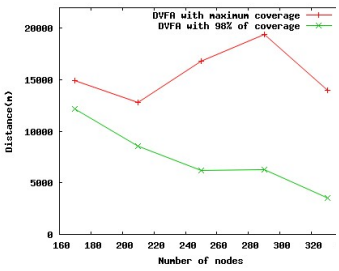

Fig. 8. Traveled distance with DVFA.

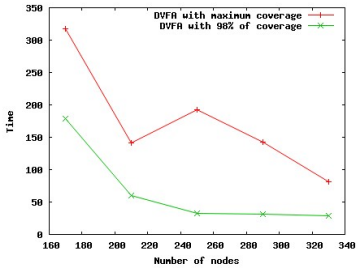

Fig. 9. Time needed with DVFA.

VI. Stability and oscillations in DVFA

In distributed redeployment protocols, an open issue is the convergence that is often slowed by node oscillations. These oscillations are harmful to node energy conservation. Some papers like [11] propose some ideas to alleviate this problem. For instance, a node will stop its movement if it travels a distance less than a predefined distance within a limited time duration. A node should also stop if it travels back and forth between almost the same positions many times. The authors propose some criteria in order to stop the nodes oscillations. However, the authors do not precise how to serialize the movements of different nodes. Moreover, the way the nodes move has a strong effect on the nodes oscillations, and the given threshold values used to limit this phenomenon are hard to fix. In this paper, we test some ideas to limit oscillations by preventing node move if some condition is met:

- if $distance(new\_position, current\_position) < \delta$, with δ a small positive real, then do not move.
- if $D_{th} - \delta \leq distance(node,\ any\ one\_hop\ neighbor) < D_{th} + \delta$ then do not move.

In figure 10, these conditions are denoted *limited distance* and *Dth-reached*, respectively. We compare the total distance traveled by nodes at the end of the simulation with and without move conditions. As a reference, we depict the total distance obtained the first time the maximum coverage is reached. We notice that with both conditions, the traveled distance increases with the number of nodes. It appears that the limited distance condition is the more efficient condition to save node energy.

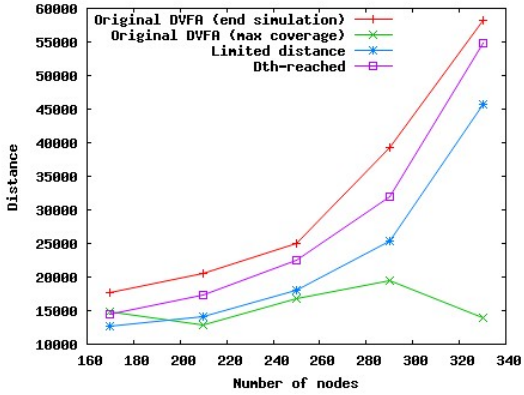

Fig. 10. Total distance traveled by nodes with/without move conditions.

VII. Conclusion

We have designed and simulated DVFA, a Distributed Virtual Forces Algorithm to ensure a uniform coverage of the monitored area and connectivity of a wireless sensor network, randomly deployed. This algorithm does not require any synchronization between sensor nodes, which are autonomous and move indepently function of the forces exerted on them by their neighborhood. By NS2 simulations, we show that DVFA ensures almost 100% coverage. Compared with the centralized version CVFA, the distance traveled by each sensor node is higher with DVFA. We observed that a large part of this distance is due to oscillations. We have tested some move conditions and shown that the limited distance condition is the more efficient condition to save node energy.

References

[1] F. Kribi, P. Minet, A. Laouiti, "Redeploying mobile wireless sensor networks with virtual forces", IFIP Wireless Days 2009, Paris, France, December 2009.

[2] G. Ciuprina, D. Ioan, I. Munteanu, "Use of Intelligent-Particle Swarm Optimization in Electromagnetics", IEEE Transactions on Magnetics, March 2002.

[3] X. Wang, S. Wang, J-J. Ma, "An Improved Co-evolutionary Particle Swarm Optimization for Wireless Sensor Networks with Dynamic Deployment", Sensors 2007.

[4] X. Wang, S. Wang, D. Bi, "Virtual Force-Directed Particle Swarm Optimization for Dynamic Deployment in Wireless Sensor Networks", ICIC (1) 2007.

[5] X. Li, N. Santoro, I. Stojmenovic, "Mesh-Based Sensor Relocation for Coverage Maintenance in Mobile Sensor Networks", UIC 2007.

[6] Y. Zou, K. Chakrabarty, "Sensor Deployment and Target Localization Based on Virtual Forces", INFOCOM 2003.

[7] Y. Zou, K. Chakrabarty, "Sensor deployment and target localization in distributed sensor networks", ACM Trans. Embedded Comput. Syst. 3(1): 61-91, 2004.

[8] E. Juergen, V. Felix, G. Reinhard, D. Falko, "Distributed Mass-Spring-Relaxation for Anchor-free Self-localization in Sensor and Actor Networks", 20th IEEE International Conference on Computer Communication Networks (ICCCN 2011), Maui, HI , USA .

[9] A. Howard, M.J. Mataric, G. Sukhatme, "Relaxation on a mesh: a formalism for generalized localization", International Conference on Intelligent Robots and Systems, (IROS 2001), Wailea, Hawaii, USA.

[10] C. Safak Sahin, E. Urrea, M. Umit Uyar, M. Conner, G. Bertoli, C. Pizzo,"Design of genetic algorithms for topology control of unmanned vehicles", International Journal of Applied Decision Sciences 2010 - Vol. 3, No.3 pp. 221 - 238.

[11] N. Heo, P. K. Varshney, "A distributed self spreading algorithm for mobile wireless sensor networks", 2003 IEEE In Wireless Communications and Networking (WCNC 2003). 2003 IEEE, Vol. 3 (2003), pp. 1597-1602 vol.3. New Orleans, LA, USA.