# Range-Free Localization for Air-Dropped WSNs by Filtering Node Estimation Improvements

Eva M. García, Aurelio Bermúdez, and Rafael Casado
Instituto de Investigación en Informática ($I^3A$)
Universidad de Castilla-La Mancha
Albacete, Spain
{evamaria.garcia, aurelio.bermudez, rafael.casado}@uclm.es

***Abstract*— Some sensor network applications involve an aerial deployment of many sensor nodes over a particular area of interest. In this context, current range-free localization proposals, based on an iterative refinement and exchange of node estimations, are not directly applicable, because they introduce a high traffic overhead. In this paper, we propose to control this overhead by means of avoiding the transmission of certain localization packets. The criterion applied by this new localization technique to filter packets is based on the amount of improvement shown after updating an estimation and the time from the last transmission. We also tune this filter in order to find an optimal trade-off between the benefit in traffic and the penalty in time.**

***Keywords-component; wireless sensor networks, localization, range-free.***

## I. INTRODUCTION

Air-dropped wireless sensor networks (ADWSNs) are an emerging research field. They consist of thousands of sensors which are carried in aerial (usually unmanned) vehicles, and deployed over the area of interest. A wide range of applications of this promising technology runs from environment monitoring [1] to support in disaster relief operations (such as wildfires [2] or earthquakes).

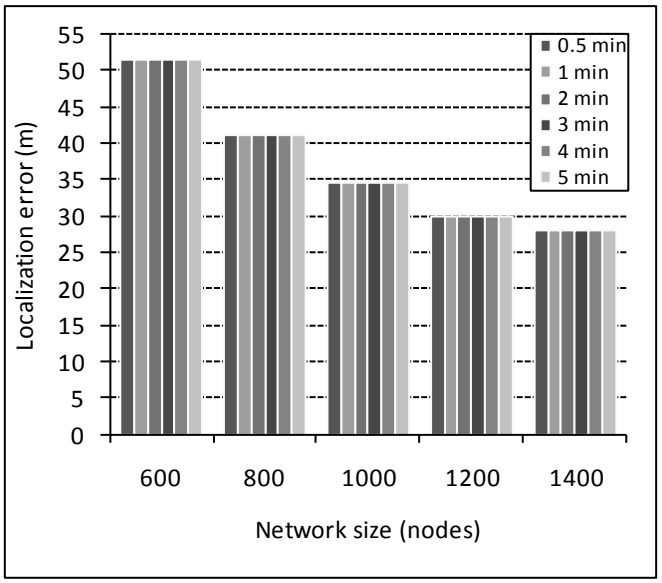
ADWSNs have specific features which distinguish them from other types of sensor networks. First, they are necessarily very dense networks containing thousands of nodes, as they have to guarantee effective and reliable terrain coverage [3]. Furthermore, the topology of the network dynamically evolves due to nodes disappearing as their battery runs out, and new nodes appearing as a consequence of several sequential deployments in the same area (to extend network service). Moreover, although they are static nodes, their position changes while they are being transported and deployed, until they finally drop to the ground. Also, nodes should be rugged enough to absorb the impact with the ground and the environmental conditions (for example a wildfire). At the same time, they must be small and lightweight enough to minimize possible material/physical damage caused during their deployment. Finally, electronic devices should be environmentally friendly, due to the impossibility of collecting them after their operation. In this respect, there are several prototypes of biodegradable devices [4].

After deployment, each network device must determine its own geographical position. This task is usually referred to as the localization process. Localization techniques may be classified into two general groups, referred to as *range-free* and *range-based* algorithms.

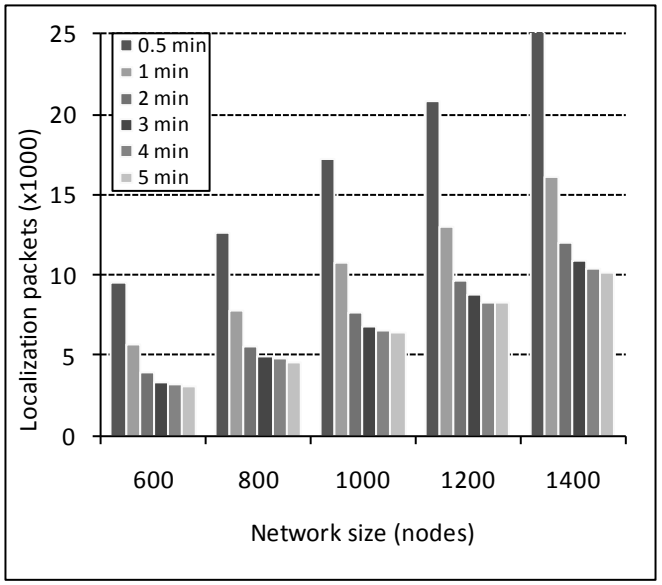
Range-based techniques estimate the position of a node on the basis of its distance from several beacon nodes. *Time difference of arrival* (TDoA) [5], *angle of arrival* (AoA) [6], and *received signal strength* (RSS) [7] are examples of methods for measuring distances between nodes. The most popular range-based location algorithm is GPS (*global positioning system*). Unfortunately, constraints of size, energy consumption, and price make it unfeasible to equip every node in a dense ADWSN with a GPS receiver. However, it may be reasonable to incorporate a GPS in just a small subset of the sensors, and use such beacons to help estimate the position of the rest of the nodes.

On the other hand, range-free techniques are based on the fact that sensor nodes are located inside the overlapping coverage area of the nodes they can hear. Assuming the existence of several beacon nodes, by means of an iterative process this area is progressively reduced, thus obtaining a more accurate estimation of the position. Some proposals employ fix-sized data structures to model the estimated localization areas. Two examples are rectangles [8] (represented by two points) and pseudo-hexagons [9] (represented by three points). Other proposals obtain more accurate areas, but progressively increase the amount of data transmitted. Some examples are convex polygons [10] (represented by up to thirty-two points) and Bézier curves [11] (represented by up to thirty points). We consider that the use of fix-sized data structures is more suitable for very dense network topologies, due to the additional overhead introduced by the latter proposals.

In this context, to support the addition of new components during the network deployment operation, nodes necessarily retransmit their location estimation periodically. Figure 1 shows the behavior of a generic range-free localization algorithm for different network sizes and transmission periods. From now on, the term transmission period means the interval of time between retransmissions of a single node. All the deployments were performed over the same area. Therefore, as network size increases, node connectivity increases (simulation




(a)




(b)

Figure 1. Localization error (a) and traffic (b), as a function of network size and transmission period, for a generic current range-free algorithm.

methodology is fully described in Section 3.1). In Figure 1(a) we can see that as network connectivity increases, average location estimations become more accurate, whereas this accuracy is not affected by the transmission period applied. Figure 1(b) shows how the traffic generated by the localization process drastically increases as network size, connectivity, and transmission frequency increase.

As shown above, dense networks natively improve the accuracy of the localization process, at the expense of dramatically increasing the traffic overhead. In this paper, we present and evaluate a range-free localization algorithm specifically designed for dense ADWSNs. To reduce the huge traffic overhead generated by hundreds or thousands of nodes executing an iterative refinement process, we propose to filter certain node transmissions. In particular, each time a node improves its estimation, it decides whether or not to retransmit on the basis of the relative improvement achieved and the time from the last transmission. This filter will be referred to as SIF (*Sent Information-based Filter*). It has been tuned in order to find a trade-off between the amount of information transmitted and the time consumed by the localization process.

The rest of the paper is organized as follows. The next section describes the behavior of range-free localization algorithms based on rectangular intersection, and introduces the proposed localization scheme. Then, Section 3 analyzes the behavior of our proposal by presenting some simulation results. Finally, in Section 4 we give our some final conclusions and outline areas for future work.

## II. SIF LOCALIZATION ALGORITHM

As mentioned in the introduction, our localization algorithm is based on the intersection of areas modeled by fix-sized data structures. In particular, for this study we have used rectangles. In this section, we first describe the foundations of these algorithms, and then we present our proposal including the SIF filter.

### *A. Principles of Rectangular Intersection*

In the literature, we can find several range-free techniques based on rectangular intersection. Basically, these proposals assume that if a node A can hear the transmission of a node B, A is located at some point inside a square that is centered at B. The side length of this square is twice the radio range of B. A seminal piece of work is the *Bounding-Box* algorithm [8], in which each node collects the position of its neighboring beacons and then obtains the intersection of the squares centered at these locations. Obviously, the result of this simple operation is a rectangle, whose center is the final estimation that the algorithms produce (see Figure 2).

Several distributed and iterative versions of the rectangular intersection technique have been proposed [12, 13, 14]. These works consider the existence of nodes not covered by the beacons. In this case, during the activation of each device, it initializes two 2-D points determining its current localization estimation rectangle ($A_C$). Beacon nodes obtain their accurate

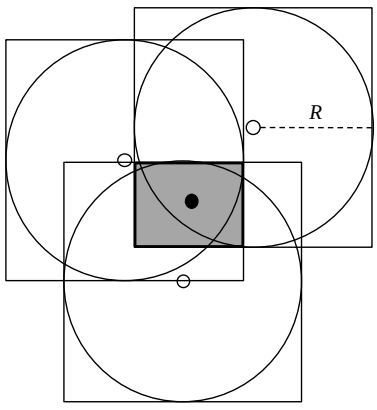



Figure 2. Rectangular Intersection. The central node estimates its position from intersection of squares representing beacons range.

position from their internal GPS receiver and, therefore, $A_C$ becomes a point (or a very small square, if we assume a margin of error). In contrast, the rest of the nodes start from an "infinite" $A_C$. Then, the iterative localization process is started by the beacons, which transmit their position to the medium. From this point, each time a node receives a localization estimation ($A_R$), it extends the received area by using the radio range. The result of this operation will be referred to as $A_{RX}$. After that, the receiving node updates its current estimation ($A_C$), by intersecting it with $A_{RX}$. Finally, the new estimation is transmitted again. One of these techniques [12] is detailed in the next table:

**Algorithm 1.** Rectangular Intersection

1: **input:** $A_R$: received area, $A_C$: current area
2: **output:** $A_C$
3: compute $A0_{RX}$: extended $A_R$
4: $A_C = A_{RX} \cap A_C$
5: send $A_C$

Figure 3 shows an example of the previous algorithm. In (a), a beacon transmits its localization rectangle (box A). A non-located node (with an infinite $A_C$) under beacon coverage receives the packet containing that localization rectangle $A_R$ (box A), and extends it by a factor equal to its radio range, obtaining $A_{RX}$ (box B). Then, the node estimates its current position $A_C = A_{RX}$ (box B). Next, in (b), the same node receives a localization rectangle transmitted by another node (box C). Again, it extends the received area and intersects it (box D) with its own (box B), obtaining a new localization rectangle (box E).

Note that the result of applying this proposal in dense ADWSNs is that it is very probable that a lot of very close nodes compute the same estimation, as they are listening to the same information. So, a large number of transmissions are redundant and may be explicitly filtered.

### *B. Our Proposal*

We shall now describe our proposal for the localization algorithm for ADWSNs. The activation of each device is performed as explained in rectangular intersection. Beacon nodes obtain their accurate position from their internal GPS receiver and the rest of the nodes start from an "infinite" rectangle.

For the execution of the localization process, we consider the existence of two modules at each network node. These modules are referred to as RCM (*rectangle computation module*) and DM (*decision module*). The RCM is activated each time the node receives a localization packet ($A_R$). Then, in the same way as the above proposals, the node refines its estimation by calculating the intersection of the current rectangle ($A_C$) with the one received, previously extending the latter by a factor equal to its radio range ($A_{RX}$). After that, the percentage difference between the previously transmitted ($A_S$) and the current rectangles is computed ($d_C$). Obviously, $A_C$ is always less than or equal to $A_S$ and, therefore, $d_C$ is positive.

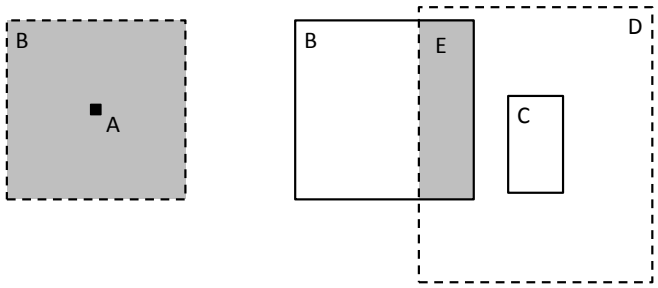



Figure 3. Example of the iterative localization process.

The reason is clear; when a node receives a new packet and two rectangles intersect, the rectangle obtained is smaller than the previous one (see Figure 3), and therefore, than the previously transmitted. On the other hand, if a node receives a packet and $A_{RX}$ does not intersect with $A_C$, then the new $A_C$ is equal to the previous one, so $d_C$=0. The next algorithm details the actions performed by the RCM module:

**Algorithm 2.** Rectangle Computation Module (RCM)

1: **input:** $A_R$: received area, $A_C$: current area, $A_S$: sent area
2: **output:** $A_C$: current area, $d_C$: difference between estimations
3: compute $A_{RX}$: extended $A_R$
4: $A_C = A_{RX} \cap A_C$
5: $d_C = (|A_S| - |A_C|) / |A_S|$

On the other hand, the DM module at each node is periodically activated in order to decide whether or not to retransmit its current estimation ($A_C$). First of all, this module compares the difference between estimations ($d_C$, previously computed) with a predetermined threshold ($d$). Then, if the threshold is exceeded, the current estimation is immediately transmitted. The transmission is also performed when a preset transmission period ($p$) has expired. In both cases, the transmission period must be reset. However, if none of the above conditions are fulfilled, the node decides not to share its estimation at this time, thus filtering the transmission. As this decision is based on the information previously transmitted by the node, we call it *Sent Information-based Filter* (SIF). Below we detail the actions performed by the DM module:

**Algorithm 3.** Decision Module (DM)

1: **input:** $d_C$, $d$: threshold difference, $t_C$: current time, $t_S$: $A_S$ sending time, $p$: transmission period
2: **output:** $A_S$, $t_S$
3: **if** $(d_C > d) \vee (t_C \geq t_S + p)$ **do**
4: send $A_C$
5: $A_S = A_C$
6: $t_S = t_C$
7: **end if**

## III. PERFORMANCE EVALUATION

After describing our proposal, in this section we evaluate its behavior by simulation. First, we present the architecture of the simulator developed for this purpose and the set of simulations performed. We then show and discuss the results obtained.

### A. Simulation Methodology

In order to evaluate our proposal, we used a simulation environment [15] developed for the EIDOS (*Equipment Destined for Orientation and Safety*) project [2], which proposes a WSN-based architecture applied to wildfire fighting operations. The environment is composed of several independent and interconnected modules, which share information by means of a global database (see Figure 4).

The core component of the system is the sensor network simulator. This module consists of a simulation engine, developed in Python, which dynamically controls a TOSSIM [16] simulation. Before starting the simulation, the engine provides each beacon with its position, modeling in this way the real behavior of a GPS receiver. During the simulation, TOSSIM is in charge of collecting several statistics, and stores them in temporary files. At the end of the simulation, the Python engine performs the storage of this information on a MySQL database.

In order to obtain realistic results, the simulator incorporates a noise and interference model and the Friis *free-space* signal propagation model [17]. The noise and interference model residing in TOSSIM uses the Closest Pattern Matching (CPM) algorithm. CPM takes a noise trace as input and generates a statistical model from it. This model can capture bursts of interference and other correlated phenomena, such that it greatly improves the quality of the RF simulation. We modeled Crossbow's IRIS mote radio XM2110CA [18], applying a transmit power of 0 dBm and a minimum received power of -88 dBm. Under these conditions, we obtain an approximate coverage range of 55 meters.

Each simulation run consists of the deployment of a sensor network over a square area of 500×500 meters. For network size, we considered 600, 800, 1000, 1200, and 1400 nodes, with an associated connectivity degree (average amount of direct neighbors) of 19.35, 25.71, 32.17, 38.91, and 45.81, respectively. Beacons represent 2% of the network nodes. Each node is deployed at a random position over the area, at a random time during the first 20 minutes of simulation. Additionally, we analyzed the impact of the dropping speed on the behavior of our proposal. In this case, we fixed the network size to 1000 nodes and considered deployment times of 5, 10, 15, 20 and 25 minutes. In all the cases, simulations conclude after 30 minutes to guarantee that the localization process finishes.

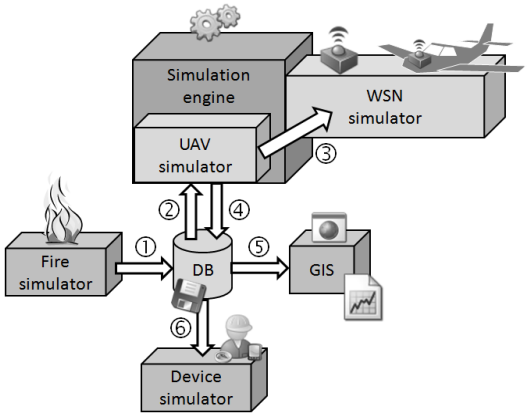
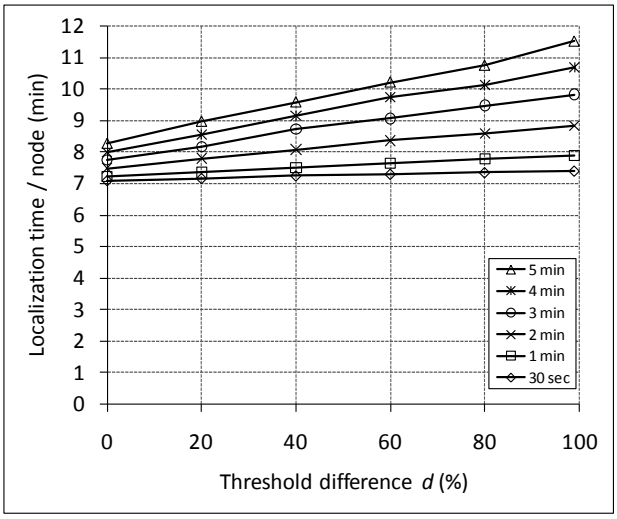


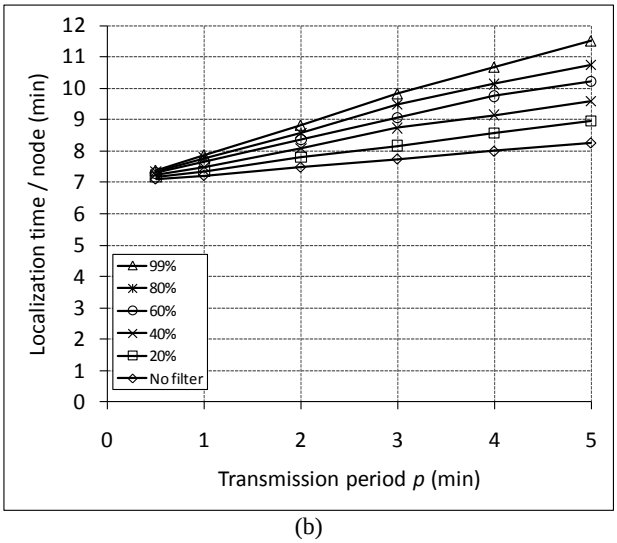

Figure 4. Architecture of the EIDOS simulation environment.

For all the scenarios described above, we varied the parameters of the SIF filter to optimize it. In particular, we used transmission periods ($p$) of 0.5, 1, 2, 3, 4, and 5 minutes; and threshold differences ($d$) of 0, 20, 40, 60, 80, and 99%.

In order to increase the accuracy of the results, each experiment were repeated 10 times for each configuration, and average values were drawn from the solution set and presented graphically. The combination of all these factors required 3240 simulations executed over 24 generic PCs.




(a)




(b)

Figure 5. Influence of the SIF filter on the behavior of the localization process. Average localization time per node against $d$ (a) and $p$ (b). Series represent $p$ (a) and $d$ (b) (network size: 1000 nodes, deployment time: 20 min).

*B. Simulation Results*

The manipulation of $d$ and $p$ parameters in the SIF filter does not affect the estimation error. Consequently, in this section we do not include results related to estimation error, as they are similar to those shown in Fig. 1(a).

Figure 5 shows the impact of $d$ and $p$ on the time consumed by a localization process executed over a network composed of 1000 nodes deployed over a period of 20 minutes. Smaller values of $d$ imply that node estimation updates are transmitted earlier, and the localization process is linearly faster, this behavior being clearer for larger values of $p$, as shown in Figure 5(a). As we can see in Figure 5(b), transmission periods shorter than $p_{min}$=0.5 minutes do not contribute significantly to speeding up the process. This is because there is a lower bound for localization time (referred to as $T_L$), which is mainly determined by the deployment time, as Figure 6 shows.

Figure 7 shows the impact of $d$ and $p$ on the traffic required by the localization process presented in Figure 5. As is obvious, the most restrictive threshold difference ($d$=99%) implies the minimum traffic overhead. Also, we can see that for transmission periods longer than $p_{max}$=5 minutes the improvement will not be significant. This provides a lower bound for localization packets ($P_L$).

Once we have established the interval [$p_{min}$, $p_{max}$], we can obtain the upper bounds for time ($T_U$) and packets ($P_U$) from the results of Figure 5 and Figure 7. Figure 8(a) presents the previous results, but expressed as a relative improvement with respect to these bounds. For localization packets, the interval [$P_L$, $P_U$] is translated to an interval of improvement of [100%, 0%]. Similarly, for localization time, the interval [$T_L$, $T_U$] is translated to an interval of (negative) improvement of [0%, –100%]. Figure 8(b) shows the aggregated improvement.

As we have seen before, the improvement in traffic increases exponentially (Figure 7), and the improvement in time decreases linearly (Figure 5). This behavior guarantees the existence of an optimal tuning that provides the maximum improvement. In particular, for the selected scenario, Figure 8 shows that the best configuration for the SIF filter is $d$=20% and $p$=2 minutes, obtaining in this case an aggregate improvement of 59%.

Figure 9 shows the aggregate improvement observed for networks with several sizes, deployed in 20 minutes. As we can see, we have obtained the same optimal configuration in all cases. Thus, we can conclude that the network size does not affect the tuning of the SIF filter.

Figure 10 shows the optimal configurations for the 1000-node network, as a function of the deployment time. In the plot, values on the left represent faster deployments than on the right. As we can see, the SIF filter is highly sensitive to deployment speed.

In very fast deployments, each node quickly improves its location estimation with the contribution of the rest of the nodes. In this case, it is better to quickly transmit periodic aggregations of several estimation improvements. As shown in the figure, SIF achieves this behavior by tuning high threshold differences combined with short transmission periods.

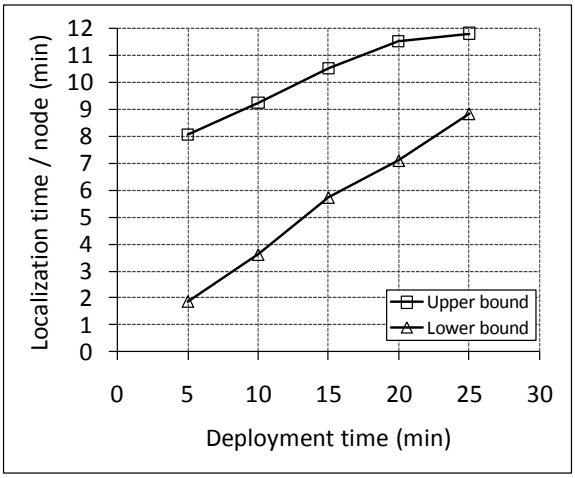
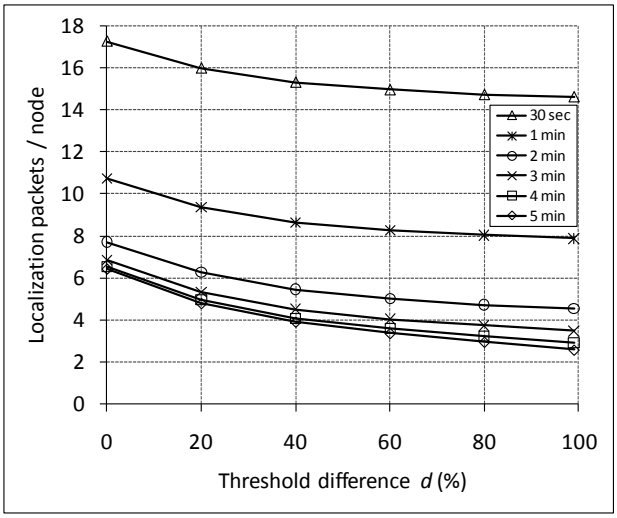
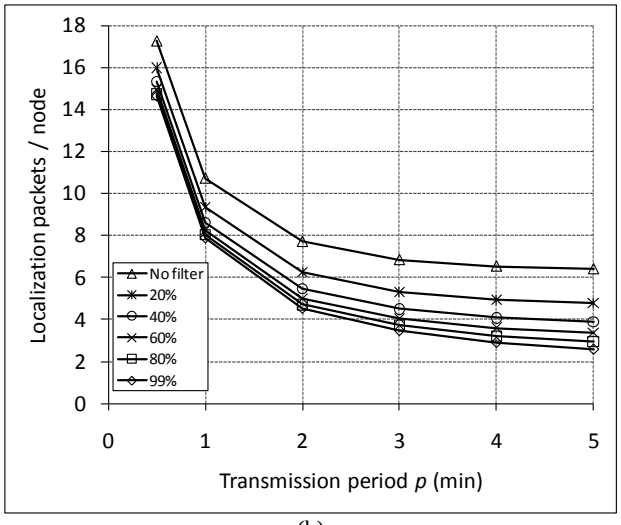



Figure 6. Localization time against deployment time (upper and lower bounds).

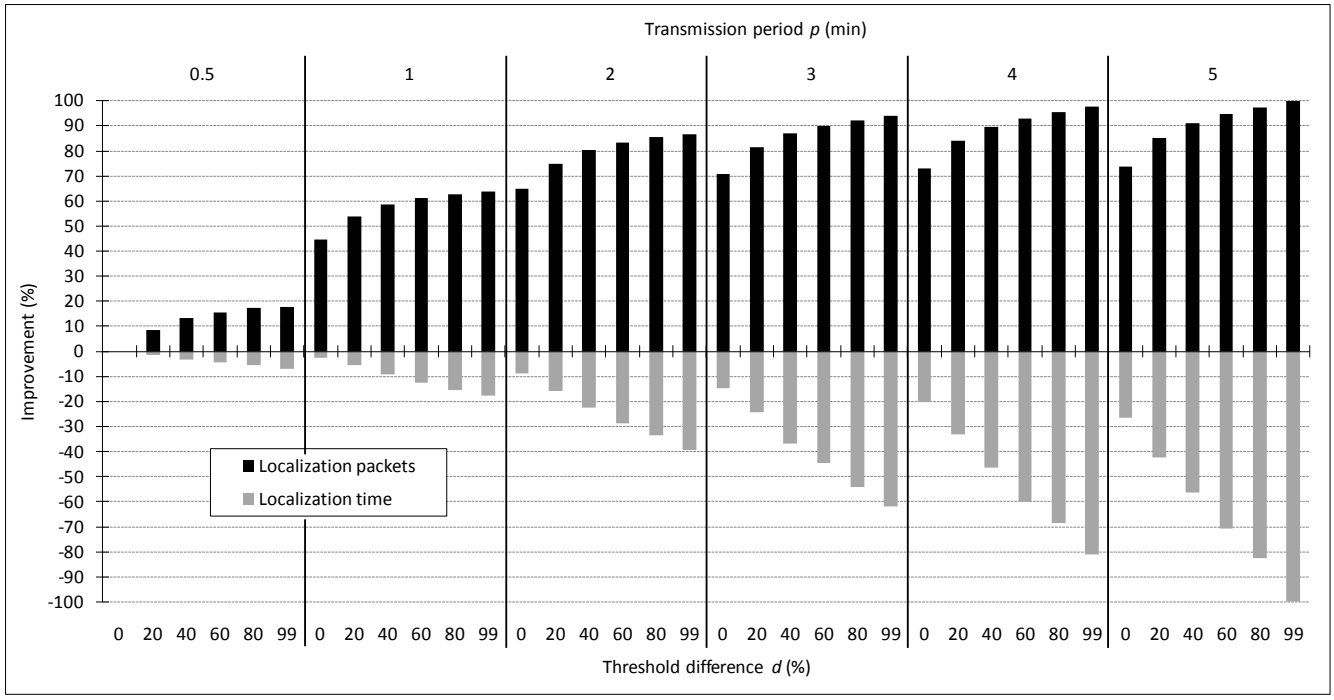



(a)

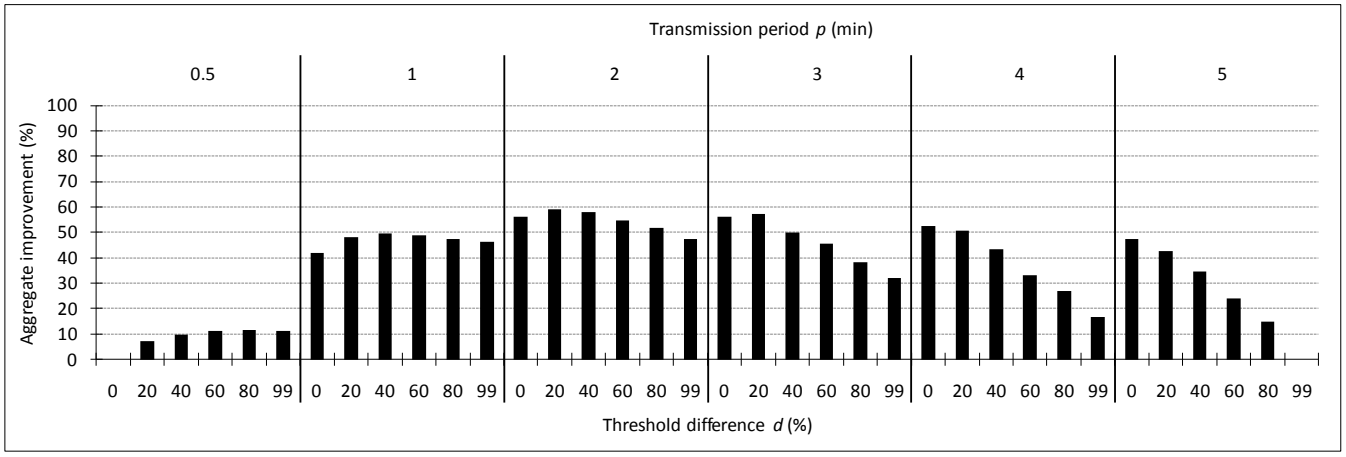



(b)

Figure 7. Influence of the SIF filter on the behavior of the localization process. Average number of localization packets per node against $d$ (a) and $p$ (b). Series represent $p$ (a) and $d$ (b) (network size: 1000 nodes, deployment time: 20 min).




(a)




(b)

Figure 8. Improvement in localization time and traffic in relation to *d* and *p* (network size: 1000 nodes, deployment time: 20 min).

On the other hand, in slower deployments, nodes may experience periods of inactivity in which their estimation remains stable. In this case, it is better for each node to avoid the retransmission of many useless copies of the same estimation, but to react quickly under new estimation improvements. SIF achieves this behavior by tuning low threshold differences combined with high transmission periods.

The global result is that nodes are able to dynamically tune the filter parameters to optimize the localization process on the basis of their detection rate of new neighbors.

## IV. Conclusions and Future Work

In this paper we have proposed a range-free localization algorithm for dense air-deployed wireless sensor networks. Our contribution is to employ a filter to reduce the huge traffic overhead inherent to the process. The filter is based on avoiding the transmission of a localization estimation if it is not sufficiently different from the previous one. We have tuned the threshold for the difference between both estimations and the transmission period in order to find the optimal trade-off between the benefit in traffic and the penalty in time. We can conclude that this tuning is dependent on the speed of deployment, but not on other parameters, such as network size.

As future work, we plan to incorporate in the nodes the ability to estimate the speed of appearance of their local nodes. This feature allows the nodes to tune the parameters *p* and *d*.

Another line of work is about the decisions taken by a node. In the SIF filter, a node takes decisions according to its own estimation refinement. We plan to allow nodes to decide whether or not to retransmit their estimation depending on

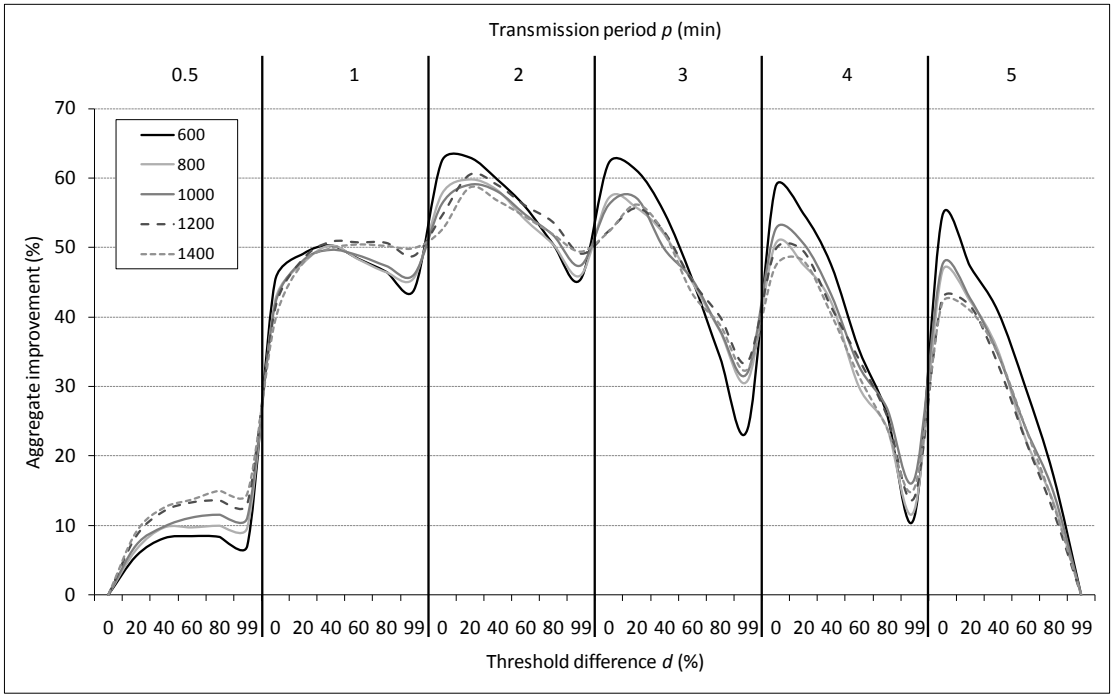



Figure 9. Aggregate improvement for different network sizes, in function of *d* and *p* (deployment time: 20 min).

whether it significantly improves the estimations transmitted from other nodes, evolving from individual decisions to collective ones.

## ACKNOWLEDGMENT

This work has been jointly supported by the Spanish MEC and European Comission FEDER funds under grants "Consolider Ingenio-2010 CSD2006-00046" and "TIN2009-14475-C04-03" and by the JCCM under grants PREG-07-25, PII1C09-0101-9476, and PII2I09-0067-3628.

## REFERENCES

[1] W.-Z. Song, R. LaHusen, R. Huang, A. Ma, M. Xu, and B. Shirazi, "Air-dropped Sensor Network for Real-time High-fidelity Volcano Monitoring," in Proceedings of the 7th Annual International Conference on Mobile Systems, Applications and Services (MobiSys09), 2009.

[2] E. M. García, A. Bermúdez, R. Casado, and F. J. Quiles, "Collaborative Data Processing for Forest Fire Fighting," in adjunct poster/demo proceedings of the 4th European Conference on Wireless Sensor Networks, 2007.

[3] C. Gamage, K. Bicakci, B. Crispo, and A. S. Tanenbaum, "Security for the Mythical Air-Dropped Sensor Network," in Proceedings of the IEEE 11th Symposium on Computers and Communications, 2006.

[4] Sensible Solutions Sweeden AB, www.sensiblesolutions.se, 2010.

[5] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan, "The cricket location-support system," in Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MobiCom'00), 2000.

[6] P. Rong and M. L. Sichitiu, "Angle of Arrival Localization for Wireless Sensor Networks," in Proceedings of the 3rd IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON'06), 2006.

[7] E. Elnahrawy, X. Li, and R. P. Martin, "The limits of localization using signal strength: a comparative study," in Proceedings of the 1st IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON'04), 2004.

[8] S. N. Simić and S. Sastry, "Distributed localization in wireless ad hoc networks," University of California at Berkeley, Technical Report No. UCB/ERL M02/26, 2002.

[9] E. M. García, A. Bermúdez, R. Casado, and F. J. Quiles, "Wireless Sensor Network Localization using Hexagonal Intersection," in Proceedings of the 1st IFIP International Conference on Wireless Sensor and Actor Networks (WSAN'07), 2007.

[10] S. Datta, C. Klinowski, M. Rudafshani, and S. Khaleque, "Distributed localization in static and mobile sensor networks," in Proceedings of the IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob'2006), 2006.

[11] R. Guha, E. Murty, and G. Sirer, "Sextant: A Unified Node and Event Localization Framework Using Non-Convex Constraints," in Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), 2005.

[12] A. Galstyan, B. Krishnamachari, K. Lerman, and S. Pattem, "Distributed online localization in sensor networks using a moving target," in Proceedings of the International Symposium of Information Processing Sensor Networks (IPSN'04), 2004.

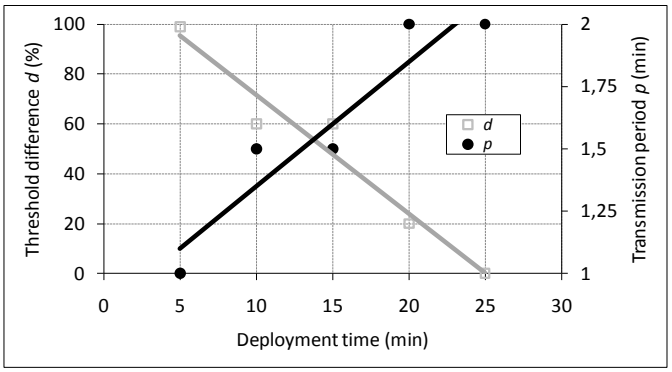



Figure 10. Optimal values for *d* and *p* in relation to deployment time (network size: 1000 nodes).

[13] A. Savvides, H. Park, and M. B. Srivastava, “The bits and flops of the N-hop multilateration primitive for node localization problems,” in Proceedings of the ACM International Workshop on Wireless Sensor Networks and Applications, 2002.

[14] J.-P. Sheu, P.-C. Chen, and C.-S. Hsu, “A Distributed Localization Scheme for Wireless Sensor Networks with Improved Grid-Scan and Vector-Based Refinement,” IEEE Transactions on Mobile Computing, 7(9), 1110–1123, 2008.

[15] E. M. García, M. A. Serna, A. Bermúdez, and R. Casado, “Simulating a WSN-based Wildfire Fighting Support System,” in Proceedings of the IEEE International Workshop on Modeling, Analysis and Simulation of Sensor Networks (MASSN'08), 2008.

[16] P. Levis, N. Lee, M. Welsh, and D. Culler, “TOSSIM: accurate and scalable simulation of entire TinyOS applications,” in Proceedings of the 1st ACM Conference on Embedded Networked Sensor Systems (SenSys’03), 2003.

[17] H. T. Friis, “A note on a simple transmission formula,” in Proceedings of the I.R.E. and Waves and Electrons, 1946.

[18] Crossbow Technology, Inc, www.xbow.com, 2010.