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

Eva M. García, Aurelio Bermúdez, and Rafael Casado

Instituto de Investigación en Informática (I3A)
Universidad de Castilla-La Mancha
02071 Albacete, Spain
{evamaria.garcia, aurelio.bermudez, rafael.casado}@uclm.es

Abstract. Many situation management applications involve an aerial deployment of a dense sensor network over the 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, due to they introduce a high traffic overhead. In this paper, we propose to control this overhead by means of avoiding the transmission of packets which do not contribute to improve the result of the localization algorithm. In particular, a node does not transmit its current position estimation if it does not significantly differ from the estimations of its neighborhood. Simulation results show that the proposed filter reduces significantly the amount of packets required by the localization process.

Keywords: wireless sensor networks, range-free localization

1 Introduction

Situation management [1] is a novel research topic in which a wide area context awareness system provides information to make better decisions. Some examples of its application may be battlefield operations or disaster response [2], [3], and [4]. Unpredictable and dynamic scenarios like these require dense real-time sensing from a large number of distributed heterogeneous information sources. A suitable approach to quickly implement the information acquisition subsystem is to deploy a wireless sensor network from the air. *Air-dropped wireless sensor networks* (ADWSNs) consist of thousands of sensing devices which are carried in aerial –usually unmanned– vehicles, and deployed over the area of interest.

After the deployment, each network device must determine its own geographical position. This task is usually referred to as *localization process*. 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 only into a small subset of the sensors, and use such *beacons* to help estimate the position of the rest of the nodes. Some localization techniques consider that a sensor node is located somewhere inside the overlapping coverage area of the nodes it can hear. Then, assuming the existence of beacons, a distributed and iterative process is executed, in which each node refines its location estimation starting from the information received from its neighborhood, and transmits the new estimation to the medium. This methodology is usually referred to as *range-free* localization.

By nature, ADWSNs are very dense networks composed of thousands of nodes to guarantee effective and reliable terrain coverage [1]. As it is well-known, dense networks improve the accuracy of range-free localization techniques. This behavior is shown in Fig. 1 ("Error" series). In this plot network density varies by deploying different amounts of nodes over the same area (simulation methodology is fully detailed in Section 4.1). However, density negatively affects traffic overhead, as "Loc packets" series shows.

Other specific feature of ADWSNs that we have to consider is their inherent network variability. Network topology dynamically evolves due to nodes that disappear as their battery is over, and new nodes that appear as a consequence of several sequential deployments in the same area (to extend service lifetime). Besides, although they may be static nodes, we must consider that their position changes while they are being transported and deployed, until they finally drop to the floor. To support this dynamism, we assume that nodes periodically retransmit their localization estimation, even after a stable situation has been achieved. This behavior differs from traditional statically deployed WSNs, in which the localization process finishes when each node obtains its final estimation. In Fig. 1, "Total packets" series shows that this periodic retransmission process introduces a huge additional overhead, if it is applied without any control.

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 thousands of nodes executing an iterative refinement process, we propose to filter node transmissions. In particular, each time a node receives a position estimation from its neighborhood, it decides whether or not to retransmit its new estimation depending on the difference between both estimations and the distance from the sending node. In this way, as we will show in the evaluation section, an important amount of transmissions from very close nodes can be avoided. This filter will be referred to as RIF (*Received Information-based Filter*).

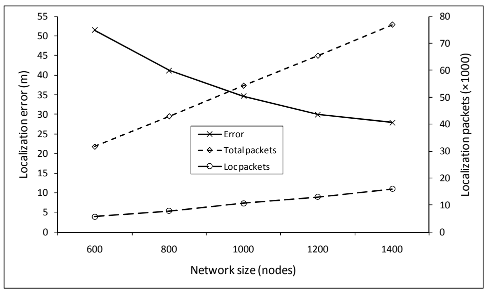


Fig. 1. Localization error and packets as a function of network size, for a generic range-free localization algorithm. "Loc packets": packets sent before the stabilization of estimations; "Total packets": packets sent after 30 minutes (retransmission period: 30 sec).

The rest of the paper is organized as follows. The next section introduces the general behavior of range-free localization algorithms. After that, Section 3 presents a localization scheme using the proposed filter. Then, Section 4 analyzes the behavior of this new technique by means of several simulation results. Finally, in Section 5 final conclusions and future work are given.

2 Range-free Localization

Localization techniques for WSNs may be classified in two general groups, referred to as *range-free* and *range-based* algorithms in the literature. Range-based techniques estimate the position of a node starting from its distance to several beacon nodes. *Time difference of arrival* (TDoA) [6], *angle of arrival* (AoA)[7], and *received signal strength* (RSS) [8]are some methods to measure distances between nodes. The most popular range-based location algorithm is GPS (*global positioning system*).

On the other hand, range-free techniques are based on the next assumption: if a node A can hear the transmission of a node B, then A is located somewhere inside an area centered at B. Some authors propose to model this area by means of a square whose side length is twice the radio range of B. A seminal work is the *Bounding-Box* algorithm [9], 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 produces.

Several distributed and iterative versions of this *rectangular intersection* technique have been proposed in [10], [11], and [12]. 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 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 [10] 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 $A_{RX}$: extended $A_R$
4: $A_C = A_{RX} \cap A_C$
5: send $A_C$
```

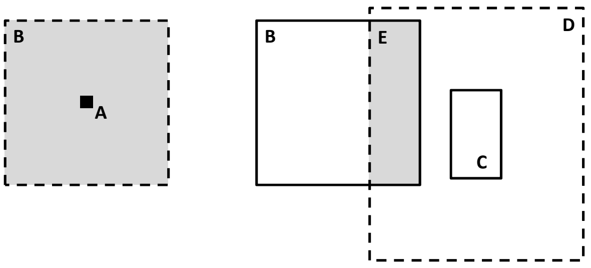


Fig. 2. Example of the iterative localization process.

Fig. 2 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).

Recently, some authors have proposed to model node coverage areas by means of hexagons [13], instead of squares, in order to reduce the additional inaccuracy introduced by this shape. In this case, the result of the area intersection operation is referred to as *pseudo-hexagon*.

Both rectangular and hexagonal intersection techniques employ fix-sized data structures to model the estimated localization areas. In particular, rectangles are represented by two points, while pseudo-hexagons can be represented by three points. Other proposals obtain more accurate estimation areas, but progressively increase the amount of data that must be transmitted. Some examples are convex polygons [14] and Bézier curves [15], represented by up to thirty two and thirty points, respectively. Obviously, 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.

3 Filtering Localization Packets

Starting from a generic rectangular intersection technique, in this section we present our proposal to avoid irrelevant retransmissions that increment the overhead without contributing to improve the accuracy. The incorporation of the filter to the localization process has been performed decomposing the algorithm into three separate modules.

A *rectangle computation module* (RCM) is responsible of refining the node estimation in the same way that the proposals described in the previous section. That is, it updates the current rectangle (A_C) by intersecting it with the one received, previously extended by a factor equal to its radio range (A_{RX}). Also, RCM computes two values that will be used later by the filter.

$d_c \in [0\%,100\%)$: fraction of A_{RX} that A_C represents. It may be obtained directly from their respective areas, due to it is satisfied that A_{RX} contains A_C.

$r_s \in [0,R]$ (R=radio range): estimation of the distance to the sender of A_R, obtained from the packet RSS[1].

The next table details the actions performed by the RCM module:

Algorithm 2. Rectangle Computation Module (RCM)

```
1: input: $A_R$: received area, $A_C$: current area
2: output: $A_C$, $d_C$: difference between
   estimations, $r_S$: distance to sending node
3: compute $A_{RX}$: extended $A_R$
4: $A_C = A_{RX} \cap A_C$
5: $d_C = (|A_{RX}| - |A_C|) / |A_{RX}|$
6: estimate $r_S$
```

A *decision module* (DM) is activated each time a node receives a localization packet. After refining the location estimation (by an invocation to RCM), this module decides whether or not to adopt the listened transmission as its own (i. e., as if this node were the sender). For this reason, our proposal is called *Received Information-based Filter* (RIF). The criterion is that the d_c and r_s values (obtained by RCM) simultaneously overcome two predetermined thresholds $d \in [0\%,100\%)$ and $r \in [0,R)$, respectively. Threshold d=0% will imply a non-filtered localization process. A value $r=R$ is excluded, due to it requires nodes covered by more than one beacon (otherwise, the process is stopped at the first iteration). Next, we detail the actions performed by the DM module:

Algorithm 3. *Decision Module* (DM)

```
1: input: $A_R$, $A_C$, $d$: threshold difference,
   $r$: threshold distance, $t_C$: current time
2: output: $t_S$: last packet sending time
3: call RCM($A_R$, $A_C$)
4: if ($d_C \le d$) $\wedge$ ($r_S \le r$) do
5:       $t_S = t_C$
6: end if
```

Finally, a *sending module* (SM) is responsible of retransmitting the current estimation if it has been too long since the last transmission was performed (or adopted). The next algorithm details the actions performed by the SM module:

[1] Some range-based localization algorithms use RSS to estimate distances among nodes, and then compute node localizations by applying multilateration. Those schemes rely on the assumption that distance measurements are accurate enough. On the other hand, our proposal employs those distances to filter transmissions, instead of using them with localization purposes. As we will show in the evaluation section, the RIF filter is highly tolerant to inaccurate measurements.

Algorithm 4. *Sending Module* (SM)

```
1: input: $t_C$, $t_S$, $p$: transmission period, $A_C$: current area
2: output: $t_S$
3: if ($t_C \geq t_S + p$) do
4:       send $A_C$
5:       $t_S = t_C$
6: end if
```

4 Performance Evaluation

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

4.1 Simulation Environment and Methodology

We have used a simulation environment [16] 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.

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 [17] 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 store them in temporal 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. We have modeled the 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 radio range (*R*) of 55 meters.

Each simulation run consists on a deployment of a sensor network over a square area of 500×500 meters. For network size, we have 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 a 2% of the network nodes. Each node is deployed in a random position over the area, in a random time during the first 10 minutes of simulation. The simulation concludes after 30 minutes, in order to guarantee that the localization process finishes. Transmission period has been set to 30 seconds.

For all the scenarios described above, we have varied the parameters of the RIF filter. In particular, we have used threshold differences (*d*) of 0%, 20%, 40%, 60%, 80%, and 99%; and threshold distances (*r*) of one quarter, one half, and three quarters of the radio range.

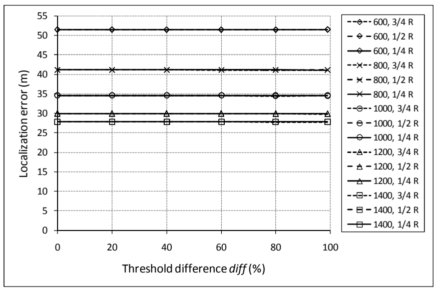


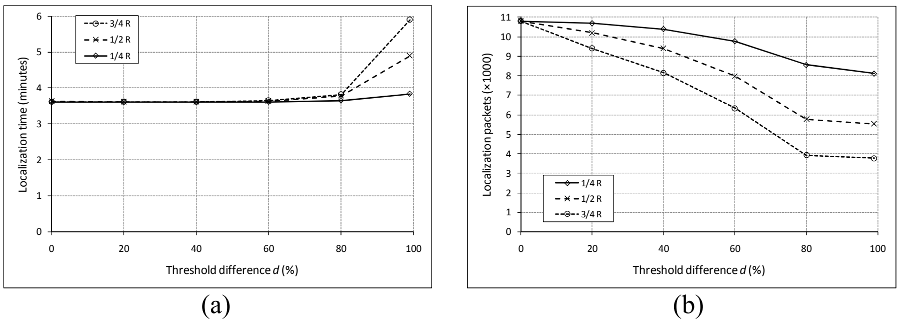
Fig. 3. Localization error as a function of the threshold difference (*d*), the threshold distance (*r*), and the network size.

With the purpose of increasing the accuracy of the results, each experiment has been repeated 10 times for each configuration, and average values have been drawn from the solution set and presented graphically.

4.2 Simulation Results

First, we check that the application of the RIF filter to the localization algorithm does not degrade the accuracy of the final estimations. Fig. 3 shows the impact of the filter parameters over the absolute error (variation between real and estimated localizations). In the figure, X-axis represents the threshold (*d*) applied to the difference between A_{RX} and A_C (d_C). Series represent several network sizes and the threshold (*r*) applied to the distance to the sending node (r_S). We can see, as stated in the introduction section, that network density contributes to reduce the average localization error. Indeed, a value *d*=0% deactivates the filter, and the obtained results match with the presented in Fig. 1. On the other hand, it is shown that, for each network size, all the combinations of *d* and *r* obtain the same result.

Fig. 4(a) shows the impact of the filter over the time employed by the process to obtain the final error estimations presented in the previous figure. X-axis represents the threshold difference (*d*), and series represent the applied threshold distance (*r*). In this case, only values for 1000-node simulations are shown. In the figure, we can see



(a) (b)

Fig. 4. Localization time and localization packets as a function of the threshold difference (*d*), grouped by threshold distance (*r*) (network size: 1000 nodes).

that only a very aggressive filtering (d>80% and r≥1/2 R) is able to significantly slow down the process.

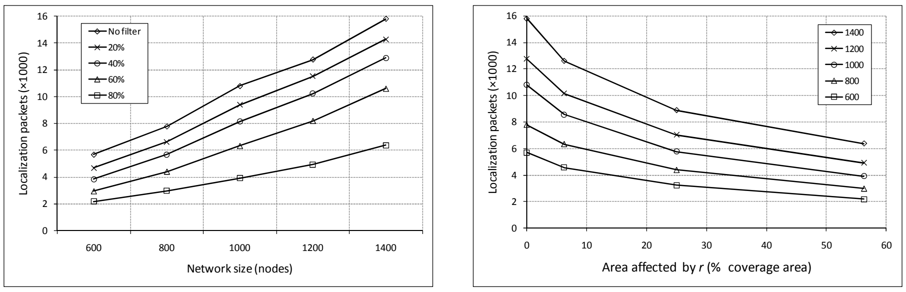
Next, Fig. 4(b) shows the impact of the RIF filter on the traffic overhead introduced by the localization processes shown in the plot (a). Again, X-axis represents the threshold difference (d), and series represent the applied threshold distance (r). Y-axis details the amount of localization packets sent by network nodes until their estimations have been stabilized.

Comparing series, we can see the expected behavior: as the range to close neighbors (under r coverage) increases, traffic overhead decreases. For each series, we can also observe the expected behavior: more restrictive values for the threshold difference (d) contribute to reduce traffic overhead. However, with very high restrictive values (d>80%), this reduction is less noticeable. A combined analysis of Fig. 4(a) and Fig. 4(b) reveals that, in this situation, relevant localization notifications are also discarded, and, therefore, the localization process is slowed down. As filtered relevant transmissions are followed by others, the global consequence is that the process execution is longer, but the traffic reduction is not effective.

A conclusion may be that a tradeoff between execution time and traffic overhead is achieved when tuning the filter with d=80% and r=3/4 R. It only increases the execution time by 5%, simultaneously reducing traffic overhead by 64%.

Fig. 5(a) shows the behavior of the filter during the localization process in function of network size. Threshold distance is fixed to 3/4 of radio range and series represent several threshold differences. A comparative analysis between series reflects the distribution of the obtained values for d_C during the simulations. We can see that the biggest traffic reduction is obtained when d varies from 60% to 80%, independently of network size.

Fig. 5(b) shows the influence of the applied threshold distance (r) on the filter's performance, in function of network size. X-axis represents πr^2 (the portion of the coverage area in which the filter is applied). The threshold difference has been fixed to 80%. We can see that as the area of application grows the relative efficacy of the filter decreases. The reason is that, although the amount of neighbors under the filter coverage increases with r, the probability to obtain higher values for d_C (different estimations) also increases. When d_C>d (fixed) transmissions are considered relevant



(a) Grouped by threshold difference (d) (threshold distance r = 3/4 R).

(b) Grouped by network size (threshold difference d = 80%).

Fig. 5. Localization packets as a function of the network size and the filtered area.

enough to not be discarded.

On the other hand, as described in Section 3, the distance r_S to the sender is obtained from the packet RSS. For this reason, it may be highly imprecise. The obtained results show that the final accuracy (estimation error) and performance (execution time) of the localization process are not affected by imprecise RSS values that may be considered as imprecise values for r_S. Also, the contribution of the filter to reduce traffic overhead is not affected for this issue. The reason is that estimated distances shorter and longer than real distances tend to offset, canceling their effect on the filter.

Next, Fig. 6 shows the impact of the RIF filter on the traffic overhead introduced by the periodic retransmissions performed once the process has converged, which guarantee the correct localization of nodes dropped in subsequent deployments. Y-axis represents the amount of localization packets sent by all nodes during 30 minutes, assuming the situation shown in Fig. 4(a), in which estimations converge after an average period of time shorter than four minutes. We can see that the application of the filter to the post-localization period is even more efficient than during the localization process. Indeed, the filter works fine for very restrictive threshold differences (d>80%), achieving an overhead reduction of 83% (d=99% and r=3/4 R).

Fig. 7(a) shows the impact of network size (and density) in traffic overhead after the localization process has finished. We can see that there is not a significant traffic reduction when tuning the filter with d≤60%. The reason is that almost all nodes are accurately located and, therefore, differences between listened estimations and the own ones are frequently substantial. The filter works fine when tuned to d=80% (as when the localization process was being executed). Indeed, it supports a very aggressive filtering (d=99%), in which only completely new information is transmitted. This minimizes traffic overhead, guaranteeing that nodes moving or being deployed will restart the localization process. Also, the filter obtains bigger reductions on traffic overhead when it is applied to very dense networks (86% for the largest simulated topology).

Fig. 7(b) shows the influence of the applied threshold distance (r) on the filter's performance once the localization process has finished. The threshold difference has been fixed to 99%. The global behavior is similar to the observed during the

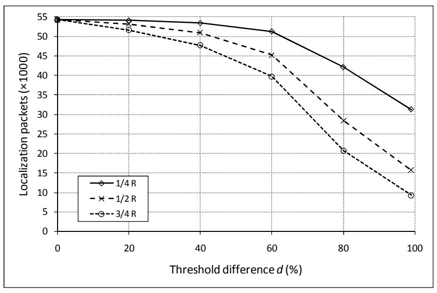
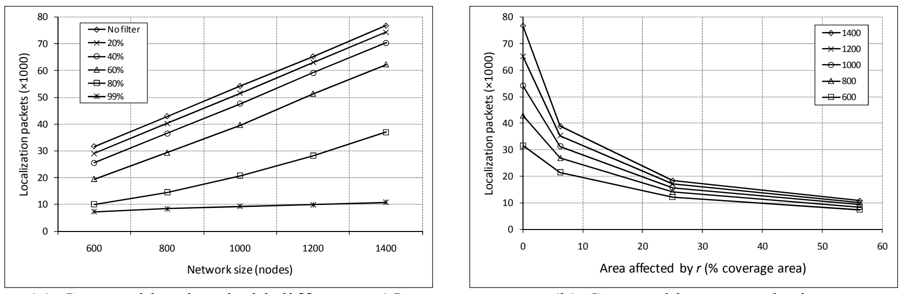


Fig. 6. Localization packets as a function of the threshold difference (d) and the threshold distance (r) (network size: 1000 nodes, execution time: 30 minutes).



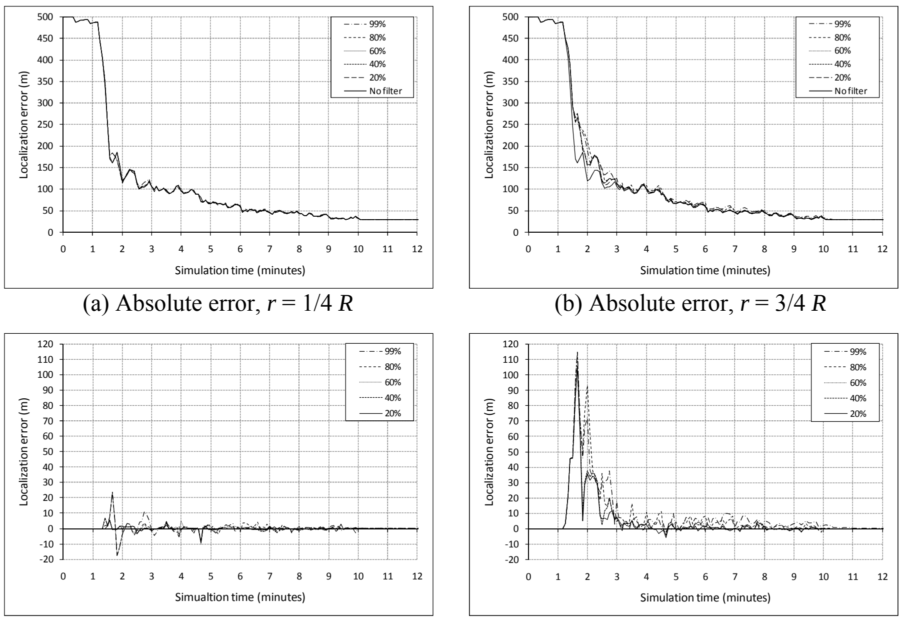
(a) Grouped by threshold difference (d) (threshold distance r = 3/4 R, execution time: 30 minutes).

(b) Grouped by network size (threshold difference d = 99%, execution time: 30 minutes).

Fig. 7. Localization packets as a function of the network size and the filtered area, grouped by threshold difference (d) (threshold distance r = 3/4 R, execution time: 30 minutes).

execution of the localization process (shown in Fig. 5(b)), but even more accentuated. The reason is that values obtained for d_C increase as estimation areas decrease, and all the nodes have achieved their respective smallest areas. The figure shows that most of the post-localization overhead (due to estimations periodically retransmitted) is eliminated for $r \leq 1/2$ R.

We have shown that the RIF filter considerably reduces the global traffic overhead, maintaining the accuracy of final estimations. Now, we study the impact of the filter in their speed of convergence, by analyzing the evolution of estimation error while the



(a) Absolute error, r = 1/4 R

(b) Absolute error, r = 3/4 R

(c) Relative error, r = 1/4 R

(d) Relative error, r = 3/4 R

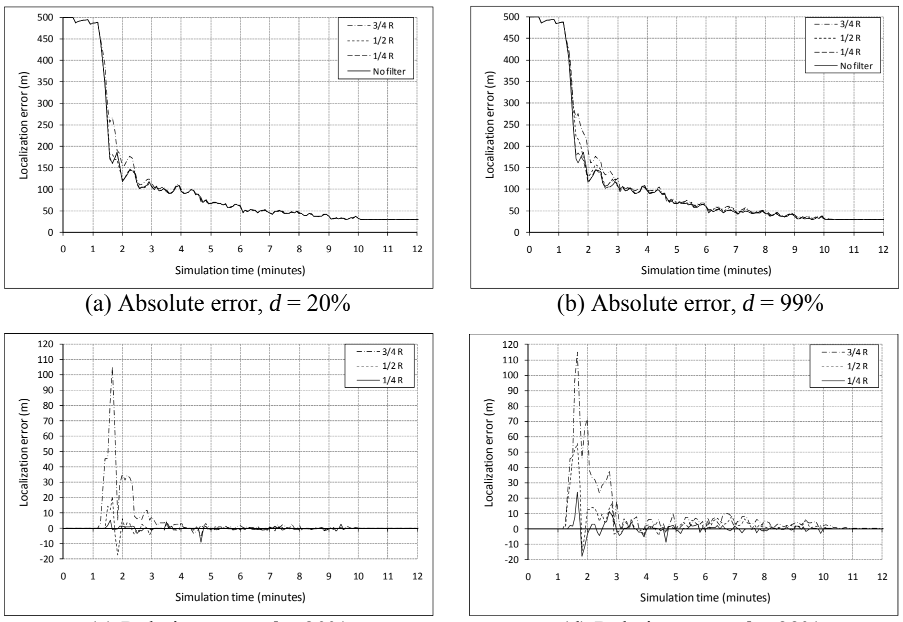
Fig. 8. Instantaneous error in function of the threshold difference (network size: 1000 nodes).

process is being executed.

Fig. 8 and Fig. 9 show the aggregated instantaneous error versus simulation time. Although an entire simulation lasts 30 minutes, the plots only show 12 minutes, due to the network is deployed during the first 10 minutes and, after that, the aggregated error stabilizes. Each series consists on a single simulation (instead of showing average values). The same deployment, composed of 1000 nodes, has been used in all cases. 20 nodes are GPS beacons, without error in their location estimation. The other 980 nodes have an initial error equal to 500 meters.

In Fig. 8, series represent the analyzed threshold differences. Plot (a) shows the absolute error obtained with the shortest threshold distance. We can see that two beacons dropped in the first minute slightly contribute to reduce the aggregated error (due to the network is not enough connected yet). The third dropped beacon produces an important reduction in the estimation errors.

To analyze the influence of the threshold difference in the speed of convergence, plot (c) shows the relative error between the series “No filter” and the rest of the series from plot (a). In general, we can observe that all of them present the same behavior and their relative errors are small. Even, sometimes, the relative error is negative, due to certain area reductions contribute to increase the error (for example when a big area centered in the real position lost a portion). Plots (b) and (d) show the results obtained for the largest threshold distance analyzed. Comparing them with plots (a) and (c), we can conclude that as *d* increases, the filter slow down the estimation updates, especially at the beginning of the process, when the network has low density. The speed of convergence is very sensible to the threshold distance



(a) Absolute error, *d* = 20% (b) Absolute error, *d* = 99%

(c) Relative error, *d* = 20% (d) Relative error, *d* = 99%

Fig. 9. Instantaneous error in function of the threshold distance (network size: 1000 nodes).

applied.

Fig. 9 is similar to Fig. 8 but, in this case, series represent several threshold distances. Left plots show results obtained for the lowest threshold difference analyzed, and rights plots show the results obtained for the highest one. Upper plots show absolute errors and lower plots relative values. From the figure we may conclude that the speed of convergence of the localization process is not affected by the threshold difference applied.

5 Conclusions and Future Work

In this paper, we have proposed a range-free localization algorithm for dense wireless sensor networks, such as ADWSNs. Our contribution is to employ a *Received Information-based Filter* (RIF) to reduce the huge traffic overhead inherent to the process. In particular, a node decides whether or not to transmit its position estimation depending on if it significantly improves the estimations received from other nodes in its neighborhood. In particular, during the localization process, the filter obtains the best performance when it only transmits high differences between listened estimations and the own one ($d_C \geq 80\%$), reducing traffic overhead to one third (approximately). Once the network is stable, the filter may be more aggressive, allowing only one transmission in a neighborhood ($d_C \geq 99\%$). In this case, traffic overhead is reduced down to 14% for very dense networks. Moreover, we have seen that all these benefits are obtained without penalizing the speed of convergence of the process and the accuracy of the obtained node localization estimations.

As future work, we plan to improve the filter, by removing the distance parameter, which requires RSS radio capabilities in network devices. Other research line may be to incorporate an internal state to the filter, which allows it to make better decisions starting from a sequence of listened estimations. Also, we plan to evaluate the impact of the RIF filter on battery consumption when thousands of network devices are being transported by aerial vehicles.

Acknowledgments. This work was supported by the Spanish MEC and MICINN, as well as European Commission FEDER funds, under Grants CSD2006-00046 and TIN2009-14475-C04. It was also partly supported by the JCCM under Grants PREG-07-25, PII1C09-0101-9476, and PII2I09-0067-3628.

References

1. Jakobson, G., Buford, J. F., Lewis, L.: Situation Management. IEEE Communications Magazine. Guest Editorial: 48(3) (2010)
2. García, E. M., Bermúdez, A., Casado, R., Quiles, F. J.: Collaborative Data Processing for Forest Fire Fighting. In Adjunct poster/demo Proc. 4th European Conference on Wireless Sensor Networks, Delft (2007)

3. George, S. M., et al.: DistressNet: A Wireless Ad Hoc and Sensor Network Architecture for Situation Management in Disaster Response. IEEE Communications Magazine, 48(3) (2010)
4. Song, W.-Z, LaHusen, R., Huang, R., Ma, A., Xu, M., Shirazi, B.: Air-dropped Sensor Network for Real-time High-fidelity Volcano Monitoring. In 7th Annual International Conference on Mobile Systems, Applications and Services (MobiSys09), Kraków (2009)
5. Crossbow Technology, Inc. http://www.xbow.com
6. Priyantha, N. B., Chakraborty, A., Balakrishnan, H.: The cricket location-support system. In 6th Annual International Conference on Mobile Computing and Networking (MobiCom'00), Boston (2000)
7. Rong, P., Sichitiu, M. L.: Angle of Arrival Localization for Wireless Sensor Networks. In. 3rd IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON'06), Reston (2006)
8. Elnahrawy, E., Li, X., Martin, R. P.: The limits of localization using signal strength: a comparative study. In 1st IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON'04), Santa Clara (2004)
9. Simić, S. N., Sastry, S.: Distributed localization in wireless ad hoc networks. University of California at Berkeley, Technical Report No. UCB/ERL M02/26 (2002)
10. Galstyan, A., Krishnamachari, B., Lerman, K., Pattem, S.: Distributed online localization in sensor networks using a moving target. In International Symposium of Information Processing Sensor Networks (IPSN'04), Berkeley, California (2004)
11. Savvides, A., Park, H., Srivastava, M. B.: The bits and flops of the N-hop multilateration primitive for node localization problems. In ACM International Workshop on Wireless Sensor Networks and Applications (WSNA'02), Atlanta, Georgia (2002)
12. Sheu, J.-P., Chen, P.-C., Hsu, C.-S.: 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)
13. García, E. M., Bermúdez, A., Casado, R., Quiles, F. J.: Wireless Sensor Network Localization using Hexagonal Intersection. In 1st IFIP International Conference on Wireless Sensor and Actor Networks (WSAN'07), Albacete (2007)
14. Datta, S., Klinowski, C., Rudafshani, M., Khaleque, S.: Distributed localization in static and mobile sensor networks. In IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob'2006), Montreal (2006)
15. Guha, S., Murty, R. N., Sirer, E. G.: Sextant: A Unified Node and Event Localization Framework Using Non-Convex Constraints. In ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), Urbana-Champaign (2005)
16. García, E. M., Serna, M. A., Bermúdez, A., Casado, R.: Simulating a WSN-based Wildfire Fighting Support System. In. IEEE International Workshop on Modeling, Analysis and Simulation of Sensor Networks (MASSN'08), Sydney (2008)
17. Levis, P., Lee, N., Welsh, M., Culler, D.: TOSSIM: accurate and scalable simulation of entire TinyOS applications. In 1st ACM Conference on Embedded Networked Sensor Systems (SenSys'03), Los Angeles (2003)
18. Crossbow Technology, Inc. http://www.xbow.com