Available online at www.sciencedirect.com
ScienceDirect
Ad Hoc Networks 6 (2008) 718–733
Ad Hoc Networks
www.elsevier.com/locate/adhoc

Monte Carlo localization for mobile wireless sensor networks

Aline Baggio *, Koen Langendoen

Delft University of Technology, Mekelweg 4, P.O. Box 5031, 2600 GA Delft, The Netherlands

Received 2 November 2006; received in revised form 10 May 2007; accepted 20 June 2007
Available online 1 July 2007

Abstract

Localization is crucial to many applications in wireless sensor networks. In this article, we propose a range-free anchor-based localization algorithm for mobile wireless sensor networks that builds upon the Monte Carlo localization algorithm. We concentrate on improving the localization accuracy and efficiency by making better use of the information a sensor node gathers and by drawing the necessary location samples faster. To do so, we constrain the area from which samples are drawn by building a box that covers the region where anchors' radio ranges overlap. This box is the region of the deployment area where the sensor node is localized. Simulation results show that localization accuracy is improved by a minimum of 4% and by a maximum of 73% (average 30%), for varying node speeds when considering nodes with knowledge of at least three anchors. The coverage is also strongly affected by speed and its improvement ranges from 3% to 55% (average 22%). Finally, the processing time is reduced by 93% for a similar localization accuracy.

Keywords: Distributed localization algorithms; Wireless sensor networks; Mobility; Monte Carlo localization; Simulations

1. Introduction

Many applications have a need for localization, be it for locating people or objects. Most of the time, data recorded from a wireless sensor only makes sense if correlated to a position. For example, the temperature recorded in a given machine room or cold-store. Similarly, many end-user programs are location-aware, for example, people would like to find the closest bus stop or mailbox, and emergency services need to find persons to be rescued. In the following, we refer to a person, object or computer coupled with a wireless sensor to be localized as an *(unknown) node*.

In both ubiquitous computing and wireless sensor networks (WSNs), localization has drawn considerable attention. The major difference between these two fields lies in the capabilities of the considered computing devices. Ubiquitous computing usually considers devices such as laptops and PDAs that are rather powerful compared to a wireless sensor. A sensor node has both a very limited memory footprint and CPU power, and energy – provided most of the time by a small battery – is a scarce resource. As such, localization algorithms for wireless sensor networks have to be efficient, both in terms of computation and power consumption. Another difference between ubiquitous

* Corresponding author. Tel.: +31 15 2787217.
E-mail addresses: A.G.Baggio@tudelft.nl (A. Baggio), K.G.Langendoen@tudelft.nl (K. Langendoen).

1570-8705/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.adhoc.2007.06.004

computing and wireless sensor networks is that laptops and PDAs have often been considered mobile while most of the existing experiments in wireless sensor networks have concentrated on static networks of sensors. At the moment, few low-cost localization algorithms exist that have been specifically designed with sensor movement in mind.

Nowadays, the most simple, off-the-shelf, mechanism to determine the location of a mobile node is to use the global positioning system (GPS) [19]. GPS offers 3D localization based on direct line-of-sight with at least four satellites, providing an accuracy up to 3 m. However, some limitations of GPS ask for alternative localization methods. First, GPS is at the moment barely usable indoors, in cluttered urban areas and under dense foliage. Second, while the cost for GPS equipment has been dropping over the years, it is still not suited for mass-produced cheap sensor boards, phones and even PDAs. Third, GPS equipment requires both hardware space and energy, which are two limiting factors for integration on miniaturized sensor boards. To overcome GPS limitations, researchers have developed fully GPS-free techniques for locating nodes as well as techniques where a few nodes, commonly called *anchors*, use GPS to determine their location and, by broadcasting it, help other nodes in calculating their own position without using GPS.

This article presents a localization algorithm for wireless sensor networks specifically designed with mobility in mind. One important factor is to let the wireless sensors benefit from mobility and not only suffer from it. Existing work [6,8,21,22,24,28] has shown that using mobile anchors in static wireless sensor networks improves the accuracy of the localization algorithm, as more nodes can benefit from the anchors' position broadcasts and as each node can hear more of these. Similarly, mobile sensors have a chance to get more information than in a fully static environment. The challenge, however, is that information in a mobile wireless sensor network gets invalidated more quickly if all the nodes are moving. In the article [12], Hu and Evans introduce a localization algorithm dealing with these different characteristics. Their approach builds upon Monte Carlo localization methods used in robotics to locate a mobile robot. In this article, we present improvements to Hu and Evans' algorithm leading to better accuracy and lower computational cost when localizing nodes.

The remaining of this article is organized as follows. Section 2 presents some background information on localization in both static and mobile wireless sensor networks. Section 3 describes both our localization algorithm and Hu and Evans' algorithm that it builds upon. Section 4 gives insight on the accuracy of the localization and efficiency of the algorithm. And finally, Section 5 concludes and gives some directions for future work.

2. Localization in WSNs

There are at the moment few localization protocols specifically designed for mobile wireless sensor networks. This section presents the work of different groups that aim at enabling localization and supporting mobility in a sensor network or in a mobile ad-hoc network.

2.1. Localization algorithms for static WSNs

Localization algorithms for static wireless sensor networks are usually classified along several axes. Some algorithms are said to be range-free or range-based and some use anchors, either one-hop or n-hop away.

A first distinction between localization algorithms deals with the use of anchor nodes. Anchors are used both as a localization aid for the nodes that need to determine their location and as a way to introduce static coordinates in a wireless sensor network. Network of nodes where no anchors are used usually establish their relative positions, possibly creating their own coordinate system. In general, the more anchors, the better the accuracy of the estimated locations. However, deploying anchors can be a tedious task and prove to be a rather expensive way of improving the accuracy of the localization algorithm.

Range-based localization algorithms use techniques such as radio signal strength indicator (RSSI) [11,23] or radio and ultrasound with angle-of-arrival [16,18] (AOA) or time-difference-of arrival [25,26] (TDOA), to measure the distance that separates an unlocalized node from an anchor. These distances, also called ranges, are sensitive to range errors, i.e., inaccuracies in the range measurements and often rely on additional hardware.

To be independent of hardware and counter range inaccuracies, researchers developed range-free methods that depend uniquely on the information – location, hop count – a node receives from its neighbors, be they anchors or regular nodes. Most range-free algorithms use mathematical [14,15,17,27] or geometrical [10,13] techniques to calculate the

position of an unlocalized node. Centroid [14] is one of the most simple range-free localization algorithms. It estimates the location of a node as the centroid of the position of the anchors heard, by taking the mean of both the x and y-coordinates of all the anchors. In this article, we concentrate on range-free algorithms that use anchors to calculate the location of unlocalized nodes.

2.2. Use of mobile anchors in static WSNs

To reduce the costs in terms of both needed hardware and deployment effort, researchers proposed the use of mobile anchors to help in localizing nodes [6,8,21,22,24,28]. A mobile anchor performs the same task as a static anchor – broadcasting its accurate location – but can take the form of a human-carried PDA or a robot. In the article [20], Parker and Langendoen propose to combine mobile anchors with a statistics-based localization algorithm. Their results show that an anchor in motion improves the accuracy in resource-poor networks where typically few anchors are available. By letting an anchor move, the percentage of nodes receiving anchor messages usually increases. A mobile anchor therefore represents many virtual static anchors. Another research aspect when using mobile anchors is to determine the best path an anchor has to follow to maximize the improvement in location-estimation accuracy.

2.3. Particle filters

In the article [12], Hu and Evans present a range-free localization algorithm for mobile sensor networks based on the sequential Monte Carlo method [5,9]. The Monte Carlo method has been extensively used in robotics [3,29] where a robot estimates its localization based on its motion, perception and possibly a pre-learned map of its environment. Hu and Evans extend the Monte Carlo method as used in robotics to support the localization of sensors in a free, unmapped terrain. The authors assume a sensor has little control and knowledge over its movement, in contrast to a robot. They target an environment where there is no hardware for obtaining ranging information, the topology of the network is unknown and most likely irregular, the density of anchors is low and both anchors and sensor nodes can move in an uncontrollable manner. The only assumption, that is, made is that the sensors or anchors move with a known maximum speed and that the radio range is common to the sensors and anchors – or is distributed together with other messages. This latter point, however, is not described by the authors.

Using the sequential Monte Carlo localization (MCL), Hu and Evans want to take advantage of mobility to improve the accuracy of localization and reduce the number of anchor nodes that are required in the network. The key idea of the sequential Monte Carlo localization is to represent the posterior distribution of a node's possible locations using a set of weighted samples. Localization happens in two steps. First, the *prediction* step leads to choosing a set of samples representing the belief of the node regarding its location. During the prediction step, a node picks random locations within the deployment area, possibly constrained by its maximum speed and the previous location samples. Second, the *filtering* step aims at removing the impossible locations from the set of samples. The filtering is done using information obtained from the environment, such as the location of the anchors in the case of a sensor node or the detection of landmarks in the case of a mobile robot. The process repeats and the sensor or robot is able to update its position estimation.

In the article [4], Dil et al. recently proposed a range-based version of the MCL algorithm introduced in [12]. By using range information from the anchors that are one and two hops away from the unlocalized node, the authors show that they can improve the accuracy of localization by (roughly) 6–10%. In the latter case, only the well-connected nodes are considered, that is, to say those that have heard information from three or more anchors. In the simulated network, the proportion of well-connected nodes was 65% when considering only the anchors that are one and two hops away from the unlocalized node. The improvement in accuracy, however, comes at a cost since the nodes have to spend more energy communicating with each others for forwarding anchor positions.

3. Localization in mobile WSNs

Apart from the experiments with the Monte Carlo localization, there are at the moment few localization protocols specifically designed with mobile wireless sensors in mind. Most of the papers presenting localization algorithms suggest that supporting mobility can be achieved by rerunning the localization algorithm after some time interval, either static or adaptable. While this is not optimal but feasible in some cases, the entire class of algorithms using information from distant nodes or

iterative approaches will suffer from severe information decay. At the time the information reaches a distant node that wants to use it, it is very likely that the whole network configuration has changed. A node will therefore always calculate an inaccurate location, not due to the lack of information or to the intrinsic inaccuracy of the calculations it uses, but due to the way its localization algorithm gathers this information.

Mobility introduces a real-time component to the localization algorithms. Wireless sensor networks are usually considered delay-tolerant [7,21]. To the contrary, mobility makes a sensor network delay-intolerant: information gathering and location calculation should happen in a timely manner, dependent on the speed of both the nodes and the anchors. This means that in a mobile wireless sensor network, methods relying on global knowledge such as calculating the number of hops or distances to all the anchors in the network are to be avoided. Similarly, a mobile node cannot really benefit from iterative localization techniques where the location estimation is refined whenever a node receives more information from the network.

Besides possible information decay, a localization algorithm deployed in a mobile wireless sensor network should be able to cope with the temporary lack of anchors. In other words, the algorithms should be able to produce a location estimate in such conditions if the application layer has a need for it. In such cases, the location estimation could easily be tagged as uncertain, providing a mean for the application to assess how much the results of the localization algorithm should be trusted.

We believe mobility should be taken into account directly when designing new localization algorithms. A wireless sensor should benefit from mobility and exploit it to improve the efficiency of localization or get a better accuracy of its position estimates. The algorithms based on Monte Carlo localization are offering such guarantees. In the following, we build upon the range-free Monte Carlo localization algorithm proposed by Hu and Evans [12] and show that by improving the way the anchor information is used, we can improve both the accuracy and the efficiency of the algorithm.

3.1. Monte Carlo localization

In [12], Hu and Evans define their localization algorithm as follows. The time is divided into discrete intervals. A sensor node relocalizes in each time interval. During the localization-algorithm initialization phase, a sensor picks a random set of N samples $L_0 = \{l_0^0, l_0^1, \ldots, l_0^{N-1}\}$, i.e., random localizations within the deployment area. From then on, the two steps, prediction and filtering, repeat. During the prediction step at time t, a sensor node generates a new set of samples L_t based on the previous set L_{t-1}. In practice, given a location l_{t-1}^i from L_{t-1}, a random location l_t^i is chosen from the disk of radius $v_{\max}$ around l_{t-1}^i, $v_{\max}$ being the maximum speed of a node. During the filtering phase, all impossible locations l_t^i are removed from the new set of samples L_t. The filtering occurs by using the position information obtained from both the one-hop and two-hop anchors. The one-hop-anchor group is composed of the anchors the sensor node heard directly. These anchors are assumed to be in the radio range r of the sensor node. The two-hop-anchor group is composed of anchors the sensor node did not hear itself but its one-hop neighbors did. These anchors are assumed to be in the range $2r$ of the sensor node but not within a radius r. In other words, MCL makes use of negative information. Note that this usually leads to an improved localization accuracy in an obstacle-free deployment area but is quite risky otherwise (see Section 4.7.3). By using information from its neighbors over the anchors they heard, a sensor can learn about the anchors that are outside its radio range.

Note that after the filtering step, there may be less samples in the set than desired. The prediction and filtering process thus repeats until the desired number of samples is reached. The location estimate of a sensor at time t is the average of all possible locations from the sample set L_t.

3.2. Monte Carlo localization boxed

Despite being quite accurate, especially in low-anchor configurations, MCL's efficiency can be improved. Drawing samples is a long and tedious process that could easily drain a lot of energy from a sensor node. Furthermore, the way MCL makes use of anchor information leaves room for improvement. Our version of the sequential Monte Carlo localization called Monte Carlo localization boxed (MCB) [1] uses steps similar to those of MCL. The major differences lie in the way we use anchor information and the method we use for drawing new samples Fig. 1 provides a summary of the MCB localization algorithm (see [2] for algorithmic details).

Initialization: At the beginning, a node has no knowledge of its location. N is a constant that denotes the maximum number of samples to maintain in a set. L_0 is the initial set of samples; Box_0 is the initial anchor box; o_0 is the initial set of observations and x_{range} and y_{range} are the maximum x and y coordinates of the deployment area, respectively:

if $o_0 = \emptyset$ **then** $Box_0 = \{(0, x_{range});(0, y_{range})\}$	Anchor box built from the coordinates of the deployment area
else $Box_0 = \{(x_{min}, x_{max});(y_{min}, y_{max})\}$ **fi**	Anchor box built from the set of one- and two-hop anchors
$L_0 = \{l_0^0, l_0^1, ..., l_0^N\}$	Set of N random locations within the anchor box Box_0

Step: Compute a new possible location set L_t based on both L_{t-1}, the set of possible locations from the previous time step $t-1$, and the new observations o_t, the position information obtained from both the one-hop and two-hop anchors between time $t-1$ and time t.

if $o_t = \emptyset$ **then** $Box_t = \{(0, x_{range});(0, y_{range})\}$	
else $Box_t = \{(x_{min}, x_{max});(y_{min}, y_{max})\}$	**Anchor box building**
fi	
$L_t = \emptyset$	
while (size(L_t) $< N$) **do**	
foreach $l_{t-1}^i \in L_{t-1}$ with $1 \le i \le N$	
$Box_t^i = \{(x_{min}^i, x_{max}^i); (y_{min}^i, y_{max}^i)\}$ for l_{t-1}^i	**Sample box building**
$R = \{l_t^i \mid l_t^i$ is selected within Box_t^i with $p(l_t \mid l_{t-1}^i) > 0\}$	**Prediction**
$R_{filtered} = \{l_t^i \mid l_t^i$ where $l_t^i \in R$ and $p(o_t \mid l_t^i) > 0\}$	**Filtering**
$L_t =$ choose $(L_t \cup R_{filtered}, N)$	
done	
done	
Location estimate $= \frac{\sum_{i=1}^{N} l_t^i}{N}$	

Anchor box building:
$Box_t = \{(x_{min}, x_{max});(y_{min}, y_{max})\}$ with (x_j, y_j) being the coordinates of the considered anchor j and n being the total number of anchors heard. We replace r by $2r$ in the following formulas when using the two-hop anchors:

$$x_{min} = \max_{j=1}^{n}(x_j - r) \quad x_{max} = \min_{j=1}^{n}(x_j + r) \quad y_{min} = \max_{j=1}^{n}(y_j - r) \quad y_{max} = \min_{j=1}^{n}(y_j + r)$$

Sample box building:
$Box_t^i = \{(x_{min}^i, x_{max}^i); (y_{min}^i, y_{max}^i)\}$ with (x_{t-1}^i, y_{t-1}^i) being the coordinates of the old sample l_{t-1}^i:

$$x_{min}^i = max(x_{min}, x_{t-1}^i - v_{max}) \quad x_{max}^i = min(x_{max}, x_{t-1}^i + v_{max})$$
$$y_{min}^i = max(y_{min}, y_{t-1}^i - v_{max}) \quad y_{max}^i = min(y_{max}, y_{t-1}^i + v_{max})$$

Prediction:
$p(l_t \mid l_{t-1}^i) = 1$ if $x_{min}^i \le x_t^i \le x_{max}^i$ and $y_{min}^i \le y_t^i \le y_{max}^i$; 0 otherwise
Assumes a node is equally likely to move in any direction with any speed between 0 and v_{max} within Box_t^i

Filtering:
$p(o_t \mid l_t^i) = 1$ if $\forall s \in S, d(l_t, s) \le r \wedge \forall s \in T, r < d(l_t, s) \le 2r$; 0 otherwise, where r is the radio range, S is the set of one-hop anchors and T is the set of two-hop anchors, $d(l_t, s)$ is the Euclidean distance between the anchor s and the sample l_t.

Fig. 1. Monte Carlo localization boxed.

The original MCL algorithm uses information about one-hop and two-hop anchors at filtering time only, for rejecting impossible samples. In MCB, we use the information about the anchors heard to constrain the area from which the samples are drawn, as explained below. Such a method is known as sampling from the optimal proposal [5]. Reducing the area to sample from has two consequences. First, we draw good samples more easily and thus faster. Drawing good samples means that we have to reject samples less often in the filtering phase, reducing thereby the number of iterations the algorithm needs to fill the sample set entirely. The second consequence is implementation dependent. Unlike the pseudo-code shown in [12], the implementation of MCL sets a bound on the

number of times a node can try to draw samples if its sample set does not contain the required number of samples yet. This boils down to avoiding that the algorithm loops endlessly if no valid sample can be drawn for a given configuration. In [12], Hu and Evans selected a sample-set size N of 50. A node tries at most twice 10,000 times to draw a sample. This happens once with a strict speed condition, drawing new samples from the disk of radius v_{max} around the old sample, and a second time with a relaxed speed condition, drawing new samples from the disk of radius $v_{max} + \text{delta}$ around the old sample. Drawing samples with a relaxed speed constraint only happens if the sample set is not full after the first series of 10,000 draws. After the 20,000 attempts, the sample set may still be not full, having less than 50 good samples. MCL does not try to fill the sample set any further. In MCB, we make sure that the sample set is as full as possible by drawing samples that do not have to be filtered and therefore do not require a redraw. In most cases, the sample set is full well before 10,000 tries. Experiences have shown that 100 attempts is ample enough to fill the sample set entirely. By ensuring that the sample set is full in 50–100 draws, a node can save precious battery power. Filling the sample set whenever possible also has a positive influence on localization accuracy over time.

The method used for constraining the area from which MCB draws samples is as follows. A node that has heard anchors – one-hop or two-hop anchors – builds a box that covers the region where the anchors' radio ranges overlap. In other words, this box is the region of the deployment area where the node is localized. We call such a box the *anchor box*. Fig. 2 shows an example of an anchor box (shaded area) in the case where three one-hop anchors were heard. For each one-hop anchor heard, a node builds a square of size $2r$ centered at the anchor position, r being the radio range. Building the anchor box consists in calculating coordinates (x_{min}, x_{max}) and (y_{min}, y_{max}) (see *Anchor box building* in Fig. 1).

In addition, in the simulation, the box-building algorithm cares for inconsistent or out-of-range boxes. In other words, for boxes where the minimum value x_{min} or y_{min} is larger than its respective maximum value x_{max} or y_{max}, the box is reset either to a box with one-hop anchors only, or to the whole deployment area. In the case where values are outside of the deployment area, for example, x_{min} is negative, we reset the value to the coordinate of the border, in our example 0.

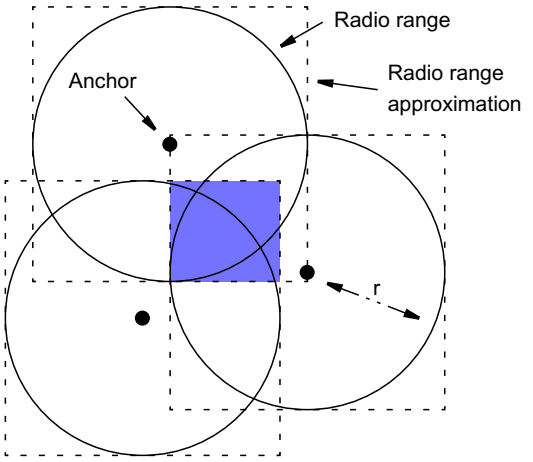

Fig. 2. Building the anchor box.

Once the anchor box is built, a node simply has to draw samples within the region it covers. Since the anchor box is a bare approximation of the radio range of the anchors, we keep a filtering step, as in the original MCL. And as in the original MCL, the prediction and filtering steps repeat until the sample set is full or until the maximum number of tries is reached.

Building an anchor box as described above is used in the case where the sample set is empty, for example, at initialization time. In the case where we already have samples, the bounding box is built with an additional constraint, namely, for each old sample l^i_{t-1} from the sample set L_{t-1}, we build an additional square of size $2 * v_{max}$ centered at the old sample, which we call *sample box*, (see *Sample box building* in Fig. 1). This updated box delimits per old sample the area a node can move in one time interval at maximum. Whenever a node has an initialized sample set but heard no anchor, we build the sample box solely based on the maximum node speed and the old samples. Box building remains a sequential process, where the anchor box is built first – and saved for subsequent uses – and updated independently for each old sample, creating thereby the sample box from which the new samples are effectively drawn.

Besides building anchor and sample boxes for drawing new samples, MCB tries to make the best possible use of all information a node received. This influences the localization algorithm in two ways. First, during the initialization phase or whenever

the sample set becomes empty, MCB allows a node to use two-hop anchor information even if it has heard no one-hop anchor. Where the original MCL makes use of two-hop anchor information only in combination with one-hop anchor information during the filtering phase, MCB allows a node to use all information it got both at prediction and filtering time. This means that a node that heard only two-hop anchors can still draw samples using these and produce a location estimate.

Second, whenever a node has heard anchors and has an already initialized sample set but has failed to fill it (entirely) with new samples, MCB reverts to solely drawing new samples from the anchor box. In other words, the sample boxes are not used anymore. Not being able to fill the sample set typically happens when too many old samples are inconsistent with the current connectivity and speed constraints. To counter old sample inaccuracy and draw new valid samples, the algorithm would need to let the node travel a too long distance, i.e., more than what could be covered with speed $v_{\max}$ in one time interval, to finally meet the connectivity constraints. In such a case, MCL would try to draw new samples with a relaxed speed constraint ($v_{\max} + \text{delta}$). Drawing solely from the anchor box in MCB is equivalent to relaxing the speed. The advantage, however, is that no delta for the allowed speed increase has to be chosen in advance as is the case with MCL.

4. Evaluation

This section presents some measurements that compare the performance of MCL and MCB. We also provide a comparison with two altered versions of MCB and with a very simple protocol: Centroid. The performance measurements are produced through simulation. In this section, we present the algorithms we used, our simulation procedure, the characteristics of the wireless sensor network we used and then provide detailed results of the various parameters we studied.

4.1. Simulated algorithms

In the following, we present the results of the simulations of five localization algorithms. We reused and extended the simulator used in [12]. First, we ran MCL as specified in [12] and presented above. MCB was implemented as described in Section 3.

For further comparison with the original MCL, we simulated two additional versions of MCB. The first one, referred to as MCF, makes a limited use of the two-hop anchors. It uses them at filtering time only, which matches what MCL does. In practice, this means that MCF constructs its anchor box based solely on the one-hop anchors (if any). It subsequently draws samples from the sample box and filters them using the connectivity constraint of both the one-hop and two-hop anchors as MCB does. In other words, we restrict the information MCF can use at box-building time.

The second additional version of MCB we tested, referred to as MCX, makes use of the two-hop anchors in both the filter and when building the anchor box as MCB does. However, the two-hop anchors are not used alone but solely in combination with the one-hop anchors (if any). This matches what MCL does too. If a node has heard no one-hop anchors but has heard two-hop anchors, this information cannot be used.

Finally, in order to compare with a well-known, simple and efficient localization algorithm, we chose to run the Centroid [14] algorithm in our simulations. Centroid calculates the position of an unknown node by computing the averages of all the x- and all the y-coordinates of the anchors heard.

4.2. Evaluation procedure

The selected localization algorithms have been tested with simulated mobile wireless sensor networks. In the following, we assume a number of nodes and anchors deployed in an obstacle-free area of 500×500 units. We thus allow all algorithms to use negative information. Both the nodes and anchors are mobile. The anchors know their location a priori, for example by using GPS. The radio range r is set to 100 units for both the anchors and the nodes.

A simulation run consists in feeding the simulator with a set of parameters such as the number of nodes in the network, the number of anchors, the maximum speed at which they are allowed move, the degree of irregularity used to model the radio communication. Time is discrete in the simulator. The simulated maximum speed of a node thus represents the maximum distance in "units" a node can move per "time unit". In the following, the maximum reachable speed per time unit is expressed as a multiple of the radio range r, in other words

v_{max}/r per time unit. The actual distance a node can cover in one time step depends on the duration, that is, chosen. The length of a time step depends on trade-offs taking into account the wished localization accuracy, energy use and radio range. A short duration provides better localization accuracy for a fast moving node but drains precious energy. For example, for a time step of one minute and a radio range r of 50 m, a fast walking human (5 km/h) would travel $1.66r$ per time step. For fast moving objects, it is of course possible to choose an adaptable time step where a node would localize more often when moving at high speeds than when moving slowly. The study of trade-offs between time-step duration, localization accuracy, information decay and energy conservation policy are left for future work.

For a given set of parameters, the simulator generates a number of random network configurations, in our case 20. For each distinct network configuration, we simulate 200 time units. The first 100 units, the nodes move without localizing. For each subsequent time unit, the nodes first localize and then move. In other words, the time freezes and we localize the whole network using a snapshot. There is no movement while the nodes are localizing. This means that message transfer is instantaneous and that the received anchor locations are still accurate when a node receives them. As such, the simulation results represent a best-case scenario where no inaccuracy is introduced due to ongoing movement, communication delays, message loss or collisions, or other anchor-location inaccuracies (i.e., GPS error). As in [12], we use a modified random waypoint mobility model [30] where each node can vary its speed at each time step before it reaches its destination. The pause time is set to 0 and the minimum node speed is set to 0.1 to avoid speed decay [31]. Finally, as suggested in [12], we use a sample set of 50 location estimations.

To analyze the simulation results, we use the following metrics. First, we analyze the localization error. As done in [12], the localization error is calculated by measuring the distance between the real location of a node and its estimated location. Second, we consider the coverage of the different algorithms, that is, the percentage of nodes that were able to calculate a location estimate. Third, we compare the processing times necessary for purely running each algorithm, thus excluding potential communications to gather anchor locations.

4.3. Network characteristics

In the following, unless stated otherwise, we use a total of 320 nodes divided as 288 regular nodes and 32 anchors. The theoretical average number of neighbors is 40. We also determine experimentally the average number of both anchors heard and neighboring nodes. These values decrease as speed increases.

The average node speed is somewhat lower than $v_{max}/2$. This effect is due to the mobility model in use in the simulator that selects a distant destination for each node and lets the node make a step towards this destination at a random chosen speed, time interval after time interval. In most cases, the final step, i.e., the last part of the trip, is traveled at a lower speed than the rest of it because the distance to the destination is smaller than the maximum distance a node can travel at maximum speed. In many cases, this last distance is very small.

The average observed anchor density is stable. The number of one-hop anchors a node can hear varies between 5.08 at speed $0.1r$ and 4.43 at speed $2r$. The average observed node density, i.e., the number of regular nodes a node can hear, is dropping slightly as the speed increases. It varies between 44.91 at speed $0.1r$ and 39.92 at speed $2r$.

Fig. 3 shows how speed influences the chance of a node to hear anchors: there are more anchor-free intervals at slow speeds. We saw that the number of anchors heard is rather stable with respect to the maximum node speed. This thus translates as follows: at speed $0.1r$, a node will remain on average 11.21 consecutive time intervals without anchors. At speed $2r$, this average has decreased down to 2.23

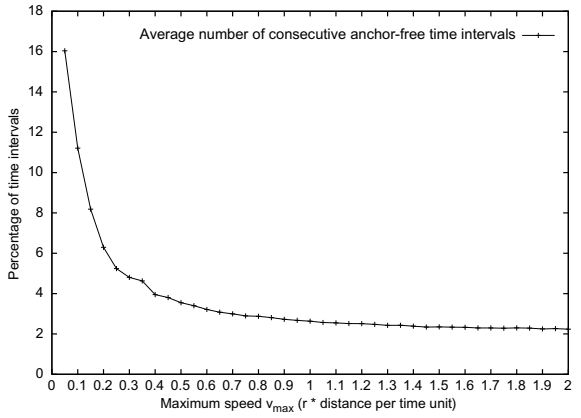

Fig. 3. Average number of consecutive anchor-free time intervals.

consecutive time intervals. The higher the number of consecutive anchor-free time interval is, the higher the localization inaccuracy as will see in Section 4.4.

Finally, on average, 78.29% of the nodes are well-connected, i.e., have heard at least three one-hop anchors. This ranges from 78.74% for speed $0.1r$ to 74.96% for speed $2r$.

4.4. Localization error

Originally, the simulator we extended (see [12]) considered the localization error for both the localized and the non-localized nodes together. The non-localized node were simply placed in the middle of the deployment area. Fig. 4 shows the localization error including the non-localized nodes. In addition, we are also considering the localization error excluding the non-localized nodes (Fig. 6) and the

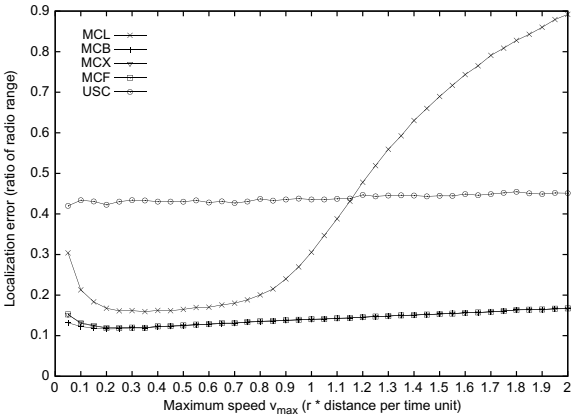

Fig. 4. Localization error (including the non-localized nodes).

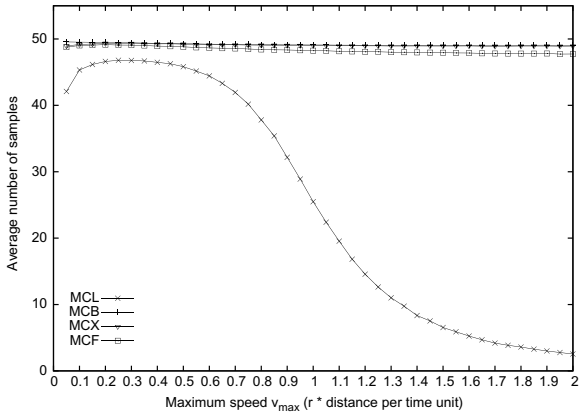

Fig. 5. Average number of samples.

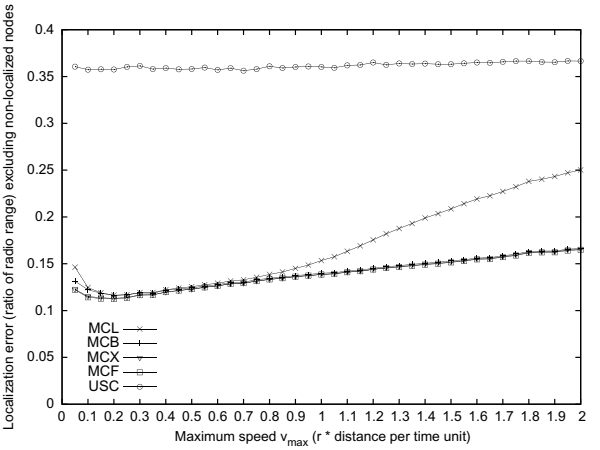

Fig. 6. Localization error (excluding the non-localized nodes).

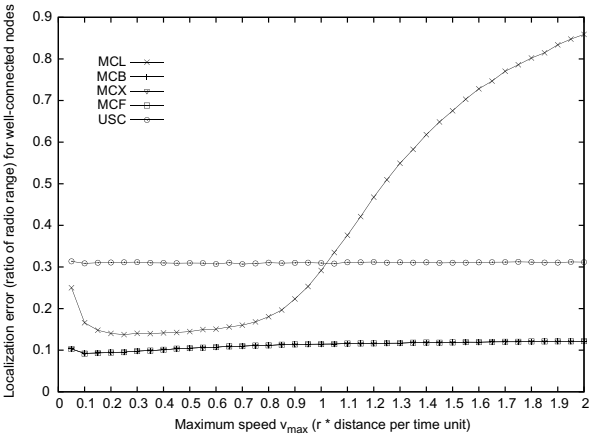

Fig. 7. Localization error (well-connected nodes).

localization error for the well-connected nodes, that is, having heard three one-hop anchors or more (Fig. 7).

4.4.1. Localization error including non-localized nodes

Fig. 4 shows the localization error including the non-localized nodes, that is, including the nodes that are placed in the middle of the deployment area because they were not able to compute a location estimate. Nodes can be non-localized for several reasons. First, they heard no anchor. This is typically the case with Centroid as it cannot produce a location estimate if no anchor is heard. In the case of Monte Carlo localization, this can happen at the beginning of the deployment when there is no previous sample set to build from. Second, in the

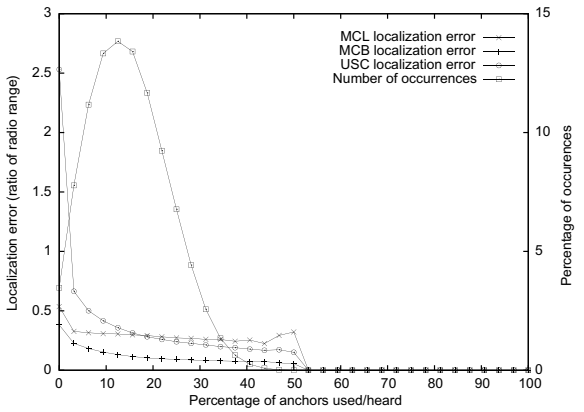

Fig. 8. Localization error per number of anchors heard (speed $1r$).

case of Monte Carlo localization, a node that has heard anchors can sometimes still be non-localized. This happens when the algorithm is not able to fill the sample set rapidly enough: the maximum number of random draws has been reached and the new sample set is still empty. This can be the case when the region to draw from is large and the area where the anchors' radio range overlap is small in comparison. Not being able to localize a node with anchors can also happen when the sample set becomes empty for some inconsistency reasons. Inconsistencies in a node's sample set generally occur after a period during which the node has heard no anchor. The new location estimations produced recursively from the old sample set gradually become less accurate as time passes and still no anchor is heard. Once an anchor is heard again, it can occur that all the new samples are rejected because they do not meet connectivity and speed constraints. Not being able to localize a node when anchors were heard is clearly unacceptable as it leads only to wasting energy and should be prevented as much as possible.

Fig. 4 shows that MCL is rather sensitive to slow and high speeds while the curve for medium speeds, i.e., for a node moving at maximum between 20% and 70% of the radio range during one time interval, remains rather flat. The localization error for the MCB variants (MCB, MCF, MCX) as well as that of Centroid (USC) are rather independent of the node maximum speed and only show a slight deterioration of the accuracy as the maximum speed increases.

The behavior of the MCL localization error with respect to the maximum speed has several causes. Slow motion gives less chances to a node to hear anchors. More precisely, the average number of anchors heard remains quite stable as the speed varies, however, the time a node can remain anchor-less is on average longer. Fig. 3 shows the average number of consecutive time intervals that a node spends without anchor at various speeds. The reason for MCL's loss in accuracy for slow speeds is clear from Fig. 3: the slower an anchor-less node moves, the less chance it has to encounter a new anchor quickly since the whole network moves only in small steps at each time interval.

At slow speeds, nodes are thus more often producing location estimates without being able to use anchor locations. This increases the inaccuracy of the set of samples over time. In the worst case, nodes are localized in the middle of the deployment area if no valid sample can be drawn once an anchor is heard again. This effect is also noticeable in the coverage results shown in Fig. 9. Note that the negative effect of slow motion was also observed by Hu and Evans in [12] for MCL.

For larger values of $v_{\max}$, such as $0.8r$ and above, the motion of the nodes allows them to hear anchors more often and this limits the decay of the sample set. However, since the distance a node can travel in a time unit is larger, the area from which the random samples are drawn also increases. This affects the accuracy in a negative way. While the average number of anchors a node hears remains rather stable and the average number of anchor-less time intervals decreases as the speed increases (Fig. 3), the average number of valid samples MCL is able to draw for high speeds considerably decreases as shown in Fig. 5. The coverage MCL achieves for high speeds is also decreasing

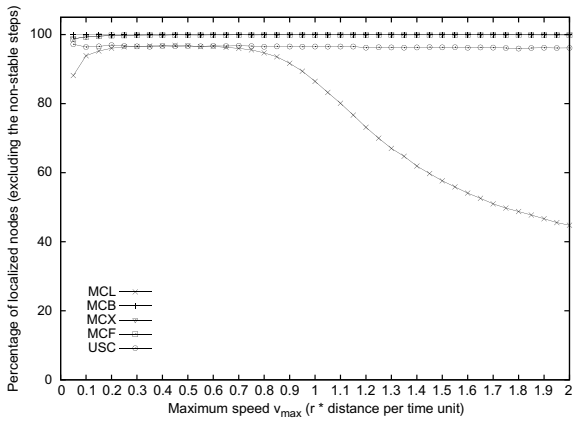

Fig. 9. Percentage of nodes localized.

(Fig. 9). The impact of the maximum speed on MCL is thus purely due to the way the algorithm produces its samples. Hu and Evans also noticed this increase in inaccuracy in [12]. They did not provide a detailed study of the coverage and average number of samples though.

The behavior of the MCB variants (MCB, MCF, MCX) is not as dramatically affected by the maximum node speed as it is the case with MCL. The main reason is that the average number of samples the MCB variants can draw is rather stable with respect to speed as shown in Fig. 5. This improves the coverage as shown in Fig. 9 and the overall accuracy. This behavior is due to the more efficient way the MCB variants draw their samples.

4.4.2. Localization error excluding non-localized nodes

Fig. 6 shows the localization error excluding the non-localized nodes. Here, the general trend is that the accuracy of each algorithm is improved. The accuracy of Centroid, for example, is improved by 7.2% on average by excluding the non-localized nodes. The improvement is 9.4% on average for MCL, ranging from 3.76% to 15.76% (speed $0.05r$) and even 63.99% (speed $2r$). The improvement in the case of MCB is negligible (0.02% on average). This is due to the fact that most nodes are localized and very few are thus excluded in Fig. 6. MCF and MCX follow a trend similar to that of MCB but show an improvement of 0.5% on average when excluding the non-localized nodes.

As a matter of fact, we observe that the improvement in accuracy of the MCB variants over MCL is mainly gained by improving the coverage. For speeds $0.2r$–$0.8r$, the improvement obtained by the MCB variants is marginal. For lower and higher speeds, the MCL curve shows the impact of the reduced average number of samples.

Comparing the curves of MCB to these of MCF and MCX, we see that the localization error is mostly identical. Only for slow speeds, for $0.05r$–$0.4r$, are MCF and MCX more accurate than MCB. This demonstrates that increasing the coverage and maintaining a high average number of samples has a price. Where MCF and MCX declare a node as non-localized, MCB keeps drawing samples and produces a location estimate. Recall that MCB can produce location estimates based solely on two-hop anchors, which MCF and MCX do not do. In other words, increasing the coverage has thus a cost in accuracy. This effect is the most pronounced for speeds up to $0.2r$.

4.4.3. Localization error for the well-connected nodes

Fig. 7 shows the localization error for the well-connected nodes, that is, to say the nodes having at least three one-hop anchors. The curves follow here the trend of Fig. 4 where the localization error including the non-localized nodes is shown. All algorithms show an improvement in accuracy when only the well-connected nodes are considered. This improvement ranges on average from 2.37% for MCB, 2.4% for MCL, and 2.6% for MCF and MCX to 12.22% for Centroid, when compared with the values from Fig. 4. All algorithms also improve their accuracy if we compare the results with those of Fig. 6 (4.9% for Centroid, 2.3% for MCB, 2.1% for MCF and MCX), except MCL (−7%). This illustrates the fact that even with well-connected nodes and thus in theory enough anchors to produce a reasonable location estimate, MCL is not always able to localize a node. This is mostly due to MCL's inefficiency when drawing samples. MCB avoids such a waste of energy and can localize a node in such cases. Remember that MCB can revert to using solely the anchor box whenever a node has failed drawing new samples from its old sample set. Reverting to the anchor box allows a node both to reinitialize a decayed sample set with fresh and valid samples and to localize based on valid (anchor) information.

4.4.4. Localization error per number of anchors heard

Fig. 8 shows the distribution of the localization error with respect to the number of anchors heard at a maximum speed of $1r$. The dotted curve shows the percentage of occurrences (right-hand y-axis), that is, to say how often nodes could heard a given number of anchors. The number of anchors is given in percent as well (x-axis). In the simulation we used a total of 32 anchors. In this case, we see that the nodes heard a maximum of 16 anchors (50%). Above 50% of anchors, there is no data and both the number of occurrences and localization error are set to zero. The figure shows that, as expected, the localization error decreases as the number of heard anchors increases. The effect is the most dramatic for Centroid which even becomes more accurate than MCL beyond 6 anchors (18.75% of anchors with an accuracy of 0.292r for MCL and 0.282r for Centroid). We also notice that the MCL curve displays a small peak around 50% of anchors

heard. This is due to an increase in the number of unlocalized nodes. This is due again to the way MCL draws samples: at speed $1r$, the area to sample from is rather large, while with many anchors, the area where the connectivity constraint are met is a rather small. This leads to sample sets with few or no samples. A similar phenomenon can be seen at slower speed with more than 50% of anchors (figures not shown here).

4.5. Coverage

Fig. 9 shows the percentage of nodes that could be localized, i.e., for which a location estimate was produced, independently of how many anchors they heard. The coverage of Centroid is on average 96.62%. That of MCL is 92.13% on average, ranging from 96.87% (speed $0.45r$) to 86.44% (speed $1r$) down to 44.81% (speed $2r$). This comes from the fact that MCL is not able to draw enough good samples from a large draw area in which the overlap of the anchors' radio range is small. This occurs in general with high maximum node speeds. In the worst case, the new sample set remains empty leading to a non-localized node. The average coverage for MCB is extremely stable with respect to maximum node speed and stays around 99.98% (variation starts from the third decimal place). The average coverage is 99.78% for MCF and 99.79% for MCX, which shows that neglecting information reduces the coverage. Both MCF and MCX are very stable too.

4.6. Processing time

Another factor positively influenced by the way MCB draws samples is the processing time, that is, to say, the time needed by the algorithm to produce a location estimate. We consider here only the computation time and not the time needed for communicating, that is, to say gathering anchor positions, listening to neighbors and forwarding anchor messages. Communication time is network-dependent and is identical for all the Monte Carlo localization variants as they all need to communicate with both one-hop and two-hop anchors. Only Centroid communicates less as it does not consider the two-hop anchors. We measured the processing time through simulation on a PC.

The processing time of the Monte Carlo localization variants depends on several factors. First, as the maximum number of samples N in the set grows, more samples have to be drawn and processing time also increases. There is of course a trade-off between the maximum number of samples in the set, the accuracy of the localization and the processing time. In [12], Hu and Evans provided an analysis of the impact of the maximum size of the sample set. We obtained similar results with our algorithm though the maximum number of samples can more easily be reduced with MCB than with MCL.

Second, the maximum number of random draws the algorithm is allowed to make also has an influence on the processing time. In the original MCL implementation, the maximum number of draws per sample is set to two times 10,000 draws for 50 samples, once with the maximum speed $v_{\max}$ and a second time with the relaxed maximum speed $v_{\max} + \text{delta}$. More precisely, MCL allows two times 10,000 draws with an uninitialized sample set, and two times 200 draws per sample (maximum 50 samples) with an initialized sample set. With similar loop values, MCB is 40–50% faster while its accuracy and coverage are better than that of MCL and its average number of samples was higher (i.e., the sample set was full more often). These tests and those that follow were conducted using a 200×200 units deployment area with one unknown node and 32 anchors. The radio range was set to 50 units and the maximum speed to 50 units per time interval.

Thanks to its simplicity, Centroid performs much faster than any Monte Carlo localization variants. It runs in 0.0095% of the time needed by MCL and 0.0168% of the time needed by MCB when both MCL and MCB are using the original random-draw parameters, i.e., 10,000 draws with an uninitialized sample set, 200 draws with an initialized sample set, 50 samples in the set at maximum.

Next, we compared MCL and MCB processing times for identical localization error when excluding the non-localized nodes. To obtain this identical localization accuracy for MCL and MCB, we varied the loop boundaries of both algorithms. We kept the maximum number of samples unchanged (50 samples). In the case of MCL, we let the algorithm draw samples 1,000,000 times, once with the maximum speed $v_{\max}$, and a second time with the relaxed maximum speed $v_{\max} + \text{delta}$. For an initialized sample set, we allow 20,000 tries twice for each of the 50 samples. With MCB, we use a maximum of 100 draws for an uninitialized sample set. For an initialized sample set, MCB uses 50 draws from the sample box and, in the case of a partially full new sample set, it allows at maximum 100 extra

draws from solely the anchor box. At speed 1r, MCL was able to produce 48.7227 samples on average and MCB 48.2153. The resulting localization accuracy was 13.9% of the radio range for MCL and 13.8% for MCB. The coverage was 98.38% for MCL and 99.96% for MCB. The relative processing time was 100% for MCL and 6.238% for MCB. This demonstrates the fact that MCB is much faster than MCL for a similar localization accuracy. Even with a slightly lower average number of samples, the coverage of MCB is better than that of MCL.

4.7. Other parameters

We studied the influence of a number of other parameters on our localization algorithm. This subsection briefly describes the results of our experiments.

4.7.1. Influence of anchor motion

In Section 4.4, we saw the influence of speed when both nodes and anchors have an identical maximum speed. In this section, we study the influence of anchor motion on the unknown nodes, more precisely, we study the impact of mobile versus static anchors.

Our simulations show that using static anchors with mobile unknown nodes has a very limited impact. With unknown nodes moving at slow speeds, the static anchors allow MCB to improve its accuracy by about 1.5% (for speed 0.2r with a radio range r of 50 units, considering only the localized nodes). As the maximum speed increases, the improvement keeps decreasing and finally using static anchors even introduces a slight inaccuracy at speed 0.8r and higher. MCL shows a similar behavior while the localization accuracy of Centroid remains slightly worse with static anchors for all maximum speeds (varying between 0.3% and 0.1% additional inaccuracy).

As the loss in localization accuracy is rather small, static anchors could be used at our advantage in protocol extensions: (1) a mobile node can deduce it is moving since there is no doubt about who is moving anymore: the node or the anchor. Anchors could even broadcast a special "tag" stating they are static. Scenarios with both static and mobile anchors could provide an attractive setting for localization; (2) static anchors fit the "static infrastructure" scenario we encounter so often in urban or office setting, thereby making deployment more realistic.

4.7.2. Influence of node and anchor density

As could be expected, increasing the total number of anchors improves the accuracy of all the studied localization algorithms. Centroid is very sensitive to the number of one-hop anchors a node can hear while the Monte Carlo-based localization algorithms still manage to localize nodes even when few anchors can be heard. The coverage of Centroid degrades dramatically in anchor-poor scenarios.

Varying the total number of non-anchor nodes also has an impact on MCL and MCB as they use two-hop anchors. Increasing the total number of non-anchor nodes increases the average number of neighbors of a node and thereby its number of two-hop anchors. This also improves accuracy, up to a certain point where it makes no difference anymore, i.e., a node then only gets "duplicate" two-hop anchors via different neighbors.

4.7.3. Influence of negative information

The original MCL algorithm makes use of negative information to filter samples obtained when considering the two-hop anchors. While this appears to work well in an obstacle-free deployment area, at least in simulation, this can lead to a larger localization error when obstacles block or dampen the radio signal of two otherwise close-by nodes. Such a scenario is illustrated in Fig. 10. We consider

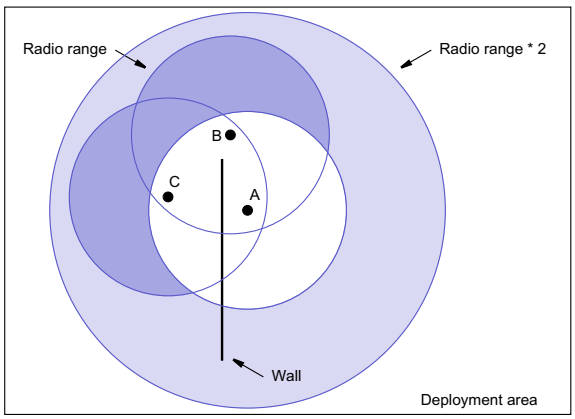

Fig. 10. Deployment area with an obstacle.

three nodes, A, an anchor; B and C, two unknown nodes. Because of the wall blocking the radio signal, node C cannot hear anchor A directly but instead gets information about the anchor location via node B. When using negative information, node C would conclude it cannot be located within the radio range of the anchor (white disk) and should be located in the donut-shaped light-gray area (two hops away). Subsequently, all the samples contained within radio range of the anchor A are rejected. In a configuration such as the one depicted in Fig. 10, this results in placing node C further away from node A than it really is and thus introduces localization inaccuracy.

Running a simple test with one wall in a 100×100-unit deployment area were almost all the nodes could hear each others revealed that using negative information has an impact on localization accuracy. We used 288 unknown nodes and 32 anchors with a radio range r of 100 units and a maximum speed of $0.5r$. As illustrated in Fig. 10, a wall ran vertically through the deployment area and did not touch the borders of the deployment area in order to get configurations where a node could not hear an anchor through the wall – in one-hop – but could reach it in two hops via a neighbor, as it is the case with nodes A, B and C.

When not using negative information, MCB consistently produced a localization error of 31.79%. The identical figures are explained by the fact that (1) all the nodes were well-connected in such an environment and (2) MCB localized all the nodes. The original MCL, using negative information, produced a localization error 34.02% of the radio range on average for the localized and non-localized nodes together or for the well-connected nodes. When considering the localized nodes only, MCL produces a localization error of 51.36%. It is worth noting that MCL managed to localized only 13.83% of the nodes while MCB localized them all, and that the non-localized nodes (placed in the center of the deployment area) actually improved the localization accuracy in the case of MCL. Finally, MCL did experience difficulties when drawing samples and its average sample-set size is only 3.3955 samples. MCB did fill its sample set more systematically and reached an average sample-set size of 49.9991.

The results of this preliminary experiment lead us to think that using negative information should be avoided in environments with obstacles that can totally block the radio signal or degrade its quality – and thus alter the radio range. This means that drawing conclusions based on the facts that a node did not hear another one are error-prone in an environment with obstacles. We are investigating this issue further with more complex obstacle configurations, such as an office floor, and various localization protocols. We also plan to study the influence of radio-range variability and determine if it causes similar problems when using negative information.

5. Conclusion and future work

Localization in wireless sensor networks is a topic that has received much interest in the past years. Most proposed algorithms concentrate on static networks of sensors with either static or mobile anchors. The problem of localizing nodes in a mobile wireless sensor network has not yet received much attention although mobility needs to be taken into account at design time.

In this paper, we presented a localization algorithm that builds upon Hu and Evans' findings [12] and that makes Monte Carlo localization more lightweight for use in wireless sensor networks. By making better use of the information a node gathers from one-hop and two-hop anchors and by restricting the area a node has to draw samples from to a (small) box, we improve the whole process of localizing. The results of simulations of our algorithm, called Monte Carlo localization Boxed, show that it allows a node to get an improved accuracy at a reduced cost. Most importantly, it ensures that a node having received information from anchors will be localized and it will not pay a high price in term of processing time and energy expenditure because of the inefficiency of the localization algorithm (random draws). Our simulation results also show that the overall coverage of the localization algorithm is improved by ensuring that the sample sets are full as often as possible.

Future work encompasses a detailed study of the energy versus accuracy trade-offs of MCB. In particular, we will study how the variations in the sample-set size, number of draws and duration of a time step affect the accuracy of localization. We will run new experiments in environments with obstacles

and study the impact of using or not using negative information.

We are also planning to deploy MCB on a test network of wireless sensors and study the behavior of the algorithm in a real-life setting. We will also make several extensions to the protocol so that it can benefit from extra information on the sensors' mobility patterns and mobility-pattern variability. This encompasses maintaining knowledge about sensors' speed and direction, possibly using additional equipment such as accelerometers and deploying it in heterogeneous networks using a mix of both mobile and static anchors.

References

[1] A. Baggio, K. Langendoen, Monte-Carlo localization for mobile wireless sensor networks, in: Second International Conference on Mobile Ad-hoc and Sensor Networks (MSN 2006), Hong Kong, China, December 2006.

[2] A. Baggio, K. Langendoen, Monte-Carlo localization for mobile wireless sensor networks, Technical Report PDS-2006-004, Delft University of Technology, June 2006.

[3] F. Dellaert, D. Fox, W. Burgard, S. Thrun, Monte Carlo localization for mobile robots, in: IEEE International Conference on Robotics and Automation (ICRA99), Detroit, Michigan, USA, May 1999.

[4] B. Dil, S. Dulman, P.J.M. Havinga, Range-based localization in mobile sensor networks, in: Third European Workshop on Wireless Sensor Networks, volume 3868 of Lecture Notes in Computer Science, Springer, Zurich, Switzerland, 2006, pp. 164–179, February.

[5] A. Doucet, N. de Freitas, N. Gordon (Eds.), Sequential Monte Carlo Methods in Practice, Springer, 2001.

[6] P. Dutta, S. Bergbreiter, Mobiloc: Mobility enhanced localization (2003), December.

[7] K. Fall, A delay-tolerant network architecture for challenged Internets, in: ACM SIGCOMM, Karlsruhe, Germany, August 2003, pp. 27–34.

[8] A. Galstyan, B. Krishnamachari, K. Lerman, S. Pattem, Distributed online localization in sensor networks using a moving target, in: Third International Symposium on Information Processing in Sensor Networks (IPSN), Berkeley, California, USA, April 2004, pp. 61–70.

[9] J.E. Handschin, Monte Carlo techniques for prediction and filtering of non-linear stochastic processes, Automatica 4 (6) (1970) 555–563, July.

[10] T. He, C. Huang, B.M. Blum, J.A. Stankovic, T. Abdelzaher, Range-free localization schemes for large scale sensor networks, in: MobiCom 2003, San Diego, CA, USA, September 2003.

[11] J. Hightower, R. Want, G.Borriello, SpotON: An indoor 3D location sensing technology based on RF signal strength, Technical Report UW-CSE 00-02-02, Springer, Seattle, February 2000.

[12] L. Hu, D. Evans, Localization for mobile sensor networks, in: Tenth International Conference on Mobile Computing and Networking (MobiCom'04), Philadelphia, Pennsylvania, USA, September 2004, pp. 45–57.

[13] D. Moore, J. Leonard, D. Rus, S. Teller, Robust distributed network localization with noisy range measurements, in: SenSys 2004, Baltimore, MD, USA, 2004, November.

[14] D.E.N. Bulusu, J. Heidenmann, GPS-less low cost outdoor localization for very small devices, in: IEEE Personal Communications Magazine, 7 (5), October 2000, pp. 28–34.

[15] R. Nagpal, H. Shrobe, J. Bachrach, Organizing a global coordinate system from local information on an ad hoc sensor network, in: Second International Workshop on Information Processing in Sensor Networks (IPSN '03), number 2634 in LNCS, Palo Alto, CA, USA, Springer Verlag, Berlin, 2003. April.

[16] D. Niculescu, B. Nath, Ad hoc positioning system (APS) using AoA, in: IEEE INFOCOM 2003, San Francisco, CA, USA, 2003 (March–April).

[17] D. Niculescu, B. Nath, DV based positioning in ad hoc networks, Telecommunication Systems 22 (1–4) (2003) 267–280.

[18] D. Niculescu, B. Nath, Error characteristics of ad hoc positioning systems (APS), in: ACM Mobihoc'2004, Tokyo, Japan, May 2004.

[19] S. Pace, G. Frost, I. Lachow, D. Frelinger, D. Fossum, D.K. Wassem, M. Pinto, The global positioning system, chapter GPS history, chronology and budgets, RAND Corporation (1995) 237–270.

[20] T. Parker, K. Langendoen, Refined statistic-based localisation for ad-hoc sensor networks, in: 47th IEEE Global Telecommunications Conference, Wireless Ad Hoc and Sensor Networks Workshop, November 2004.

[21] P.N. Pathirana, N. Bulusu, A.V. Savkin, S.K. Jha, Node localization using mobile robots in delay-tolerant sensor networks, IEEE Transactions on Mobile Computing 4 (3) (2005) 285–296, May–June.

[22] R. Peng, M.L. Sichitiu, Localization of wireless sensor networks with a mobile beacon, in: First IEEE Conference on Mobile Ad-hoc and Sensor Systems (MASS 2004), Fort Lauderdale, FL, USA, October 2004.

[23] R. Peng and M.L. Sichitiu, Robust probabilistic constraint-based localization for wireless sensor networks, in: Second Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON'05), Santa Clara, CA, USA, September 2005.

[24] N.B. Priyantha, H. Balakrishnan, E.D. Demaine, and S. Teller, Mobile-assisted localization in wireless sensor networks, in: INFOCOM 2005, Miami, FL, USA, March 2005.

[25] A. Savvides, C.-C. Han, M.B. Strivastava, Dynamic fine-grained localization in ad-hoc networks of sensors, in: Proceedings of the 7th annual international conference on Mobile computing and networking (MobiCom'01), ACM Press, 2001, pp. 166–179.

[26] A. Savvides, H. Park, M.B. Srivastava, The bits and flops of the n-hop multialteration primitive for node localization problems, in: First ACM International Workshop on Wireless Sensor Networks and Application, Atlanta, GA, USA, September 2002.

[27] S. Simić, S. Sastry, Distributed localization in wireless ad hoc networks. Technical Report UCB/ERL M02/26, EECS Department, University of California, Berkeley, 2002.

[28] K.-F. Ssu, C.-H. Ou, H.C. Jiau, Localization with mobile anchor points in wireless sensor networks, IEEE Transactions on Vehicular Technology (May) (2005) 1187–1197.

[29] S. Thrun, D. Fox, W. Burgard, F. Dellaert, Robust Monte Carlo localization for mobile robots, Artificial Intelligence 128 (1–2) (2001) 99–141, May.
[30] J. Yoon, M. Liu, B. Noble, Random Waypoint Considered Harmful, IEEE INFOCOM, San Fransciso, CA, USA, 2003, March–April.
[31] J. Yoon, M. Liu, B. Noble, Sound Mobility Models, ACM MobiCom, San Diego, CA, USA, 2003, pp. 205–216, September.

Aline Baggio is currently a postdoctoral fellow at the Delft University of Technology. Her research focuses on localization algorithms for wireless sensor networks and support for sensor mobility. From 1999 to 2004, she worked at the Vrije Universiteit in Amsterdam. There, she concentrated on adding support for mobility and replication in distributed location services. An extension of this work has been applied to the World-Wide Web under the form of distributed redirection. She received her Ph.D. in Computer Systems in 1999 from the University Pierre et Marie Curie in France.

Koen Langendoen is an associate professor in the Parallel and Distributed Systems group at Delft University of Technology, The Netherlands. He earned an M.Sc. in computer science from the Vrije Universiteit, Amsterdam in 1988 and a Ph.D. in computer science from the Universiteit van Amsterdam in 1993. His research interests include system software for parallel processing, wearable computing and, in particular, wireless sensor networks. Since 2002, he heads a sensor networking group at Delft working on localization algorithms, MAC protocols, and various pilot applications