Wireless Netw
DOI 10.1007/s11276-008-0096-3

Monte Carlo localization of wireless sensor networks with a single mobile beacon

Rui Huang · Gergely V. Záruba

Abstract One of the most important tasks in sensor networks is to determine the physical location of sensory nodes as they may not all be equipped with GPS receivers. In this paper we propose a localization method for wireless sensor networks (WSNs) using a single mobile beacon. The sensor locations are maintained as probability distributions that are sequentially updated using Monte Carlo sampling as the mobile beacon moves over the deployment area. Our method relieves much of the localization tasks from the less powerful sensor nodes themselves and relies on the more powerful beacon to perform the calculation. We discuss the Monte Carlo sampling steps in the context of the localization using a single beacon for various types of observations such as ranging, Angle of Arrival (AoA), connectivity and combinations of those. We also discuss the communication protocol that relays the observation data to the beacon and the localization result back to the sensors. We consider security issues in the localization process and the necessary steps to guard against the scenario in which a small number of sensors are compromised. Our simulation shows that our method is able to achieve less than 50% localization error and over 80% coverage with a very sparse network of degree less than 4 while achieving significantly better results if network connectivity increases.

R. Huang · G. V. Záruba (✉)
Computer Science and Engineering Department,
The University of Texas at Arlington, 416 Yates,
300NH, Arlington, TX 76019, USA
e-mail: zaruba@uta.edu

R. Huang
e-mail: rui.x.huang@gmail.com

Keywords Wireless sensor networks · Monte Carlo sampling · Localization · Particle filter

1 Introduction

Wireless sensor networks (WSNs) are special types of ad hoc networks, where large populations of small sensor-enabled nodes form the network. Many researchers view WSNs as ad hoc networks where energy efficiency is of utmost importance. Generally, sensors are seen to be spread over a target area, and the network is formed among the sensors in an ad hoc manner. Once deployed the sensors ubiquitously serve as the "eyes and ears" monitoring the deployment area (e.g., for temperature, movement, or pressure). WSNs have been attracting considerable research interest due to the large number of military and civilian applications already existing and foreseeable in the near future.

Location discovery is emerging as one of the more important tasks as application, transport, network, and data link layers would all benefit from physical location information. Researchers have observed and shown that (semi-) accurate location information could greatly improve the performance of tasks such as routing, energy conservation, data aggregation, and maintaining network security. For instance, algorithms such as LAR [10], GRID [15], and GOAFR+ [12] rely on the location information to provide more stable routes during unicast route discovery. The availability of location information is also required for geocast (multicast based on geographic information [9]) algorithms such as LBM [11], GeoGRID [16], and PBM [17]. To minimize power consumption, the GAF algorithm [27] uses location information to effectively tune the network density by turning off certain nodes at particular instances. Furthermore, in [7], the authors have shown that wormhole attacks can be effectively prevented when location information is available.

A direct way of obtaining location information is to install a global positioning system (GPS) receiver on each sensor. However, this is impractical currently as GPS receivers are still relatively expensive, power-hungry, and require clear line of sight (i.e., restricted to unobstructed outdoor scenarios) to several earth-bound satellites. WSN devices are imagined as small as possible while operating on restricted power budgets, thus it may not be feasible for them to integrate GPS receivers.

2 Related works

The localization problem is commonly formulated as follows. Given a network graph $G = (V,E)$ where a subset of the nodes $\{V\}$ are location aware *anchor* nodes (also called *beacons*) $\{V_{gps}\}$ such that $|V_{gps}| \leq |V|$, the objective of the localization algorithm is to find the locations of *non-anchor* nodes $\{V\} - \{V_{gps}\}$.

Localization methods can be classified based on the sensory data required for their operation. The availability (and of equal importance, the accuracy) of the sensory data largely dictates how well the localization method performs to an extent where it is pointless to compare different localization methods when their sensory data are incompatible. The following classification can be made based on the sensory data:

1. *Range-free (or connectivity-only)*: At a minimum, a deployed sensor can detect the connectivity to its neighbors; a number of localization methods rely on such connectivity information only. The Centroid method [3] estimates the location of an unknown node as the average of its neighbor's locations. The APIT method [6] estimates the node location by isolating the area using various triangles formed by anchor nodes. The DV-Hop method [18] counts the hop numbers to anchors and uses them as crude estimates for distances. Range-free methods require no additional hardware, but they generally only work well when networks are dense. Sparse networks by nature contain less connectivity information and are thus more difficult to localize accurately.
2. *Range-based*: A node can be localized if the distances (i.e., the ranges) to three known locations are obtained. The distances can be obtained, e.g., by measuring RSSI (received signal strength indication) or TOA (time of arrival). Some of range-based methods include the ad hoc positioning system (APS) methods such as DV-Distance and Euclidean proposed in [18, 20]. In [23], ranging data are exchanged between the neighbors to refine the initial location guess. Comparing to range-free methods, range-based methods give more accurate location estimates when ranging data is reliable. However, depending on the deployment environment, ranging techniques based on RSSI tend to be error-prone and strong filtering is required. The ranging error could ultimately destroy the localization accuracy if it is allowed to propagate through the network unboundedly.
3. *Bearing-based (or angle of arrival)*: Simulation studies in [4] show that when Angle of Arrival (AoA) of signals can be computed in addition to distance measurements, the localization accuracy and coverage can be drastically improved. An enhancement of the APS algorithm is proposed in [19] and [21] that works with AoA readings.
4. *Hybrid*: A combination of the above techniques can be employed to form a hybrid method. For instance, a hybrid method is proposed in [1] that uses both APS and MDS (multi-dimensional scaling).

The earlier localization algorithms have primarily targeted both static and mobile ad hoc networks (MANETs). A comparison between more well-known algorithms such as DV-Hop, Euclidean and Multi-lateralization is performed in [13]. The comparison is done in the context of specific constraints of sensor networks, such as error tolerance and energy efficiency; results indicate that there is no single algorithm that performs "best" and that there is room of further improvement.

A number of centralized localization methods have also been proposed based on semidefinite programming (SDP) [2] and linear programming (LP) [14]. In these algorithms, sensory data between the neighbors such as distance and angle are collected into a centralized place. Such data is then used as constraints to a global optimization problem to be solved using SDP or LP. These algorithms give the best possible localization result simply because they assume that the entire constraint set is available. However, such centralized approach can be difficult to implement in a distributed environment like WSNs.

Since WSNs are special cases of MANETs, localization algorithms designed for MANETs work in WSNs as well. However, many characteristics unique to WSNs make it practically infeasible to apply MANET localization algorithms; some of the WSN specific challenges include:

1. *Limited computing capacity*: Sensor networks consist of nodes with very limited computing capacity. In addition, sensors are likely to have restricted and limited energy sources further restricting the total amount the computation that can be performed by them. This limited local computing capacity of sensor

networks precludes any localization algorithms that require extensive computation at the sensor nodes.

2. *Limited sensory capacity*: Studies have shown that more accurate localization can be performed if nodes are capable of measuring distances and angles to their neighbors. However for such operations additional hardware might be required (e.g., to obtain the distance readings from neighbors, nodes have to be equipped with sensors that measure RSSI or TOA). AoA sensors are currently technologically infeasible as they mostly rely on programmable directional antennas. Availability of these sensory capacities just for the localization purpose may increase the complexity and the cost of sensor nodes. Above and beyond, it may be a strong and restricting assumption that all sensor nodes have the same sensory capacity; thus, it is desirable to devise localization algorithms with less reliance on a particular sensory capacity.
3. *Stricter security requirements*: One of the driving applications of sensor networks is to monitor hostile battlefields for enemy activities. In such adversary environments, it is essential for localization algorithms to perform their tasks securely. For localization algorithms that rely on sensor collaboration, it is important that the results are not drastically skewed when a small number of sensors are compromised. Unfortunately, most existing localization algorithms do not consider such security requirement.
4. *Deployment requirement*: WSN nodes are generally deployed in an ad hoc manner; when sensors are "spread" over a battlefield, it is often difficult to exercise any control over the resulting topology. Thus, in the context of localization it is difficult to obtain an ideal beacon placement strategy. For instance, it is well-known that most algorithms perform better when anchors are placed around the edge of the network. However, such beacon deployment is often difficult to achieve. Network mobility is another factor; unlike MANETs, sensor networks are generally viewed as stationary once deployed (although some researchers envision mobile WSNs). Thus, it is often sufficient for the localization algorithms to ignore the mobility requirement of MANETs.

An alternative method is to localize sensors using a mobile beacon. In this method, a mobile beacon travels through the deployment area while broadcasting its location along the way. Sensors localize themselves by monitoring information coming from the beacon. A straight-forward technique using the above method is described in [25], where sensors are required to receive at least three communications with the same RSSI reading from the beacon. Given that the same RSSI readings imply similar distances to the beacon locations, the physical sensor location can be derived using simple geometric functions. Computation-wise, this method is simple making it suitable for resource-limited sensors. However, it requires the beacon to directly pass by the ranging area of the sensor. In addition, in most cases, the beacon has to pass by the sensor twice because the sampling positions of the beacon where the three RSSI readings are taken should not be on the same line. This method also assumes that errors are insignificant in the RSSI to distance translation.

Instead of computing the location directly, a probabilistic approach may be taken; here sensor location is viewed as a probability distribution over the deployment area. In [24], sensors measure a series of RSSI readings from the mobile beacons and localize themselves by a sequential update process to the probability distributions of their locations. Each sensor starts with a uniform distribution covering the entire deployment area. As the beacon passes through, the distribution is updated to fit the received RSSI readings (using a propagation model). The method is further improved in [22] by adding the negative informdation (that is, the information that the beacon is out of range) as well as RSSI readings from the neighbors. These probabilistic methods provide much improved location estimates when compared to deterministic approaches due to the fact that they explicitly consider the impreciseness of location estimates. For each sensor, probabilistic methods do not output a single estimated location; instead, they give an area where a sensor might reside along with the likelihood of such estimate. On the other hand, probabilistic methods have the drawback of being complex. For a deployment grid of n by n units, the time and space complexity is $O(n^2)$. As the sensors at present time have very limited resources it is difficult to implement these methods directly for the large deployment. Indeed, the experimental results shown in [24] are performed on pocket PCs, which are much more powerful than common sensors.

A similar method of localizing the networks using a mobile beacon is presented in [5]. Instead of the actual probability distribution, the possible sensor locations are represented with a bounding box. As the beacon passes by, the area contained by the bounding box is progressively reduced as positive and negative information are processed. The bounding box method drastically simplifies the probability computation, making it possible to implement this method on sensors. However, such large simplification has its side-effects in that it sacrifices the preciseness of the distribution for its simplicity as it is not possible to describe multiple possible locations with a single box. There is also an additional problem when noise from the ranging sensors is considered. This method may work well when ranging error is minimal, however when erratic errors are present (which is inevitable when using RSSI ranging), there might be situations where no bounding box exists to satisfy all readings.

Recognizing various drawbacks of previous methods, in this paper we propose a hybrid approach that uses a single mobile beacon to localize nodes in WSNs. We propose a probabilistic framework where sensory readings are processed using a Monte Carlo filtering technique to compute the probability distributions as densities of particle samples. In our method, the localization is performed entirely at the mobile beacon instead of the sensors. During localization, the sensors are only responsible for relaying the sensory data and localization results. For a sensor deployment over a large area, unmanned vehicles or airplanes can serve as mobile beacons. Localization can be performed by a computer onboard or at the base station after the mobile beacon relaying the sensory data back via a secure channel. Our method also supports sensors of different sensory capacity such as ranging, AoA or connectivity only, and allows them to co-exist in the same network. Our Monte Carlo filtering technique allows those different types of sensors to collaborate during localization.

3 Monte Carlo sampling framework for WSN location estimation

By sampling a random process with a random variable and maintaining the outcomes (samples) of the random variable, the probability density of the random process can be approximated. In our case the random variable is two dimensional and takes a value according to the probability of a node's physical location. For instance, when a node is not yet localized, the best estimate about its location's random variable is that it is uniformly distributed and thus the location distribution can be approximated as a set of randomly placed samples over the entire deployment area. When a node receives a transmission from the mobile beacon, it will obtain a distance reading d from the beacon as well as the beacon's location; the random variable now is modified to a uniform distribution over a circle around the beacon and thus samples will be placed on a ring around the beacon with radius d. When a second distance reading from the beacon (residing at a different location) is overheard, the random variable (optimally) becomes a discrete random variable with two equal probabilities at the intersection points of two rings (the sample set will concentrate on the intersection points of two rings). The exact location can be found when the third broadcast is overheard, assuming all three beacon locations are not collinear. Overall, there are advantages and disadvantages of using sample points to represent probability densities. While preciseness of actual distributions is somewhat lost by approximating them with sample points, the real advantage of using sample points is that we free ourselves from the complexity of individual distributions (as all distributions would have to be described by mathematical formulas even in the presence of errors). To represent more complex distributions, or to represent distributions more precisely, only the population of the sample set has to be increased. Compared to the bounding box method in [5], sample sets give much better resolution and can deal with distributions with several significant "modes". (For instance, the bounding box cannot precisely describe the actual distribution of the two intersecting rings as described above.)

3.1 Localization using a mobile beacon

Without the loss of generality in the following discussion we will assume that all sensors and the mobile beacon have the same transmission range. (Our model is easily extended to cases where this is not true; however for the sake of a simple discussion we will keep the previous assumption.) A pair of sensors or a sensor and the beacon can overhear each other if they are in each other's range, i.e., when they are *neighbors*.

Algorithm 1 presents a pseudo-code for our localization algorithm. Sensors are responsible for collecting the range sensory data from neighbors, i.e., this data contains at least one-hop connectivity information. The same data will also include the range or AoA readings from the neighbors if such sensors are available. Such data can be easily collected by observing "Hello" messages from neighbors. The sensors will await the arrival of the beacon, and transmit the collected one-hop sensory data to the beacon. All localization processing is done at the beacon and the result is transmitted back when a sensor is successfully localized.

Algorithm 1 Localization procedure executed at each sensor

```
form a observation set $R_v = (...r_w...)$, where $r_w$ is an observation from neighbor $w$
send $R_v$ to beacon upon request
receive localization result ($id$, $x$, $y$, $var$)
if $id = v$ then
    ($x$, $y$) is the location of the sensor, and $var$ is the variance
else
    forward ($id$, $x$, $y$, $var$) to the neighbor $id$
end if
```

The localization algorithm at the mobile beacon is listed as Algorithm 2. Location distributions for sensors are stored as sample sets at the mobile beacon. We assume that the movement of the beacon is defined by a bacon movement model. For each new location of the beacon, there are two types of observations that are useful to localize the sensor: (i) positive observations in which a sensor can hear the beacon; (ii) negative observations where a sensor is out of range. The beacon updates the localization distributions for both positive and negative observations. To reduce the amount of processing, we define a critical region for each sensor v, so that the location distribution will only be updated when the beacon arrives at the critical region. For connectivity-only observations, it is more useful to update the distribution as the beacon just enters or is about to leave the sensor range. Range-based observations are useful when the movement trajectory is not collinear. AoA-based observations do not have many restrictions as long as the beacon does not pass directly over the sensor. Based on the above observations, we define the critical region as a pre-defined outer ring of the maximum transmission range of the beacon. The location distribution will be updated when a sensor just "arrives" in the critical region or when the beacon changes its moving direction.

Algorithm 2 Localization procedure executed at the mobile beacon

```
$\hat{R} \leftarrow$ storage of all observation sets
$X_v \leftarrow$ sample set of sensor $v$, initialized uniformly
$X_{beacon} \leftarrow$ sample set of the beacon reflecting the current location
move over the deployment area based on a movement model
for all sensor $v$ in $\hat{R}$ such that there is new negative observation $N_v$ do
    $X_v \leftarrow$ FilterUpdate($X_v$, $X_{beacon}$, $N_v$)
end for
if the current location is in the critical area of sensor $v$ OR beacon just
  changed direction then
    request $R_v$ from $v$
    update $R_v$ in $\hat{R}$
    $X_v \leftarrow$ FilterUpdate($X_v$, $X_{beacon}$, $R_v$)
    if $var(X_v) < T_{var}$ then
        send localization result ($v$, $x$, $y$, $var$) to $v$, where ($x$, $y$) is the
  expected location
    end if
    for all $v$'s neighbor $w$ do
        $X_w \leftarrow$ FilterUpdate($X_w$, $X_v$, $R_w$)
        if $var(X_w) < T_{var}$ then
            send localization result ($w$, $x$, $y$, $var$) to $v$, where ($x$, $y$) is the
  expected location
        end if
    end for
end if
```

The beacon calls the UpdateFilter procedure to update the location distribution of sensor v based on the sensor's latest observation data R_v as well as a reference distribution X_{beacon}. The reference distribution X_{beacon} is simply the current location of the beacon. After the location distribution of sensor v is updated, the beacon will also update the location distribution of v's neighbors based on the one-hop observation data between the pair of neighbors. Here, we use X_v as the reference distribution. This later step helps to increase the localization performance as it allows the sensors to be localized even when the beacon does not pass by them directly. Our simulations show that using the one-hop observation data reduces the average estimation error up to 50% when the beacon can only spend limited amount of time over the deployment area.

The beacon decides whether a sensor is localized based on the variance of the location distribution. If the variance reduces below a pre-define threshold T_{var} (as defined by the user or the application), the sensor is considered localized. The mode of the particle density distribution is used as the estimated location, which is transmitted back to the sensor. The variance of its particle distribution can also be sent back to serve as an indicator of the estimation quality, with smaller variances indicating better estimates. Note that the threshold T_{var} could vary depending on the application that utilizes the location information. It has been previously noted that precise locations are not essential for some applications [26]. In those cases, a larger threshold can be chosen so that better localization coverage can be obtained.

3.2 Classic Monte Carlo sampling-based Bayesian filtering

For each of the sensors, starting from a uniform distribution of randomly-place samples, the sample distribution is progressively updated using Bayesian filtering. This section outlines the theoretical background behind Bayesian filtering. Let us envision a grid system superimposed over the entire tracking area, and let the state s_t be the location of the node to be tracked in the grid system at the time t. Our goal is to estimate the posterior probability distribution, $p(s_t|d_1,...,d_t)$, of potential states—s_t, using certain measurements, $d_1,...,d_t$. The calculation of the distribution is performed recursively using a Bayes filter:

$$p(s_t|d_1,\ldots,d_t) = \frac{p(d_t|s_t)\cdot p(s_t|d_1,\ldots,d_{t-1})}{p(d_t|d_1,\ldots,d_{t-1})}$$

Assuming that the Markov assumption holds, i.e., $p(s_t|s_{t-1},\ldots,s_0,d_{t-1},\ldots,d_1) = p(s_t|s_{t-1})$, the above equation can be transformed into the recursive form:

$$p(s_t|d_1,\ldots,d_t) = \frac{p(d_t|s_t)\cdot\int p(s_t|s_{t-1})\cdot p(s_{t-1}|d_1,\ldots,d_{t-1})ds_{t-1}}{p(d_t|d_1,\ldots,d_{t-1})},$$

where $p(d_t|d_1,\ldots,d_{t-1})$ is a normalization constant. In the case of localizing a mobile node from RSSI measurements, the Markov assumption requires that the state contains all available information that could assist in predicting the next state and thus, an estimate of the non-random motion parameters of the nodes is required as part of the state description. Starting with an initial, prior probability distribution, $p(s_0)$, a system model, $p(s_t|s_{t-1})$, representing the motion of the mobile node (the mobility model), and the measurement model, $p(d|s)$, it is then possible to drive new estimates of the probability distribution over time, integrating one new measurement at a time. Each recursive update of the filter can be broken into two stages:

Prediction: Use the system model to predict the state distribution based on previous readings

$$p(s_t|d_1,\ldots,d_{t-1}) = \int p(s_t|s_{t-1})\cdot p(s_{t-1}|d_1,\ldots,d_{t-1})ds_{t-1}$$

Update: Use the measurement model to update the estimate

$$p(s_t|d_1,\ldots,d_t) = \frac{p(d_t|s_t)}{p(d_t|d_1,\ldots,d_{t-1})}p(s_t|d_1,\ldots,d_{t-1})$$

To address the complexity of the integration step and the problem of representing and updating a probability function defined on a continuous state space (which therefore has an infinite number of states), the approach presented here uses a sequential Monte Carlo filter to perform Bayesian filtering on a sample representation. The distribution is represented by a set of weighted random samples and all filtering steps are performed using Monte Carlo sampling operations. Since we have no prior knowledge of the state we are in, the initial sample distribution, $p_N(s_0)$, is represented by a set of uniformly distributed samples with equal weights, $\{(s_0^{(i)}, w_0^{(i)})|i \in [1,N], w_0^{(i)} = 1/N\}$ and the filtering steps are performed as follows:

Prediction: For each sample, $(s_{t-1}^{(i)}, w_{t-1}^{(i)})$, in the sample set, generate a random replacement sample according to the system (mobility) model $p(s_t|s_{t-1})$. This results in a new set of samples corresponding to $p(s_t|d_1,\ldots,d_t)$:

$$\{(\tilde{s}_t^{(i)}, w_t^{(i)})|i \in [1,N], w_t^{(i)} = 1/N\}$$

Update: For each sample, $(\tilde{s}_t^{(i)}, w_t^{(i)})$, set the importance weight to the measurement probability of the actual measurement, $\tilde{w}_t^{(i)} = p(d_t|\tilde{s}_t^{(i)})$. Normalize the weights such that $\sum_i \eta\cdot\tilde{w}_t^{(i)} = 1.0$, and draw N random samples for the sample set $\{(\tilde{s}_t^{(i)}, \eta\cdot w_t^{(i)})|i \in [1,N]\}$ according to the normalized weight distribution. Set the weights of the new samples to $1/N$, resulting in a new set of samples $\{(s_t^{(i)}, w_t^{(i)})|i \in [1,N], w_t^{(i)} = 1/N\}$ corresponding to the posterior distribution $p(s_t|d_1,\ldots,d_t)$.

3.3 System model

To apply the classic Monte Carlo sampling-based Bayesian filtering approach described in the previous section, we need to provide a system model and a measurement model. For the system model, we simply assume that at any point in time the node moves with a random velocity drawn from a Normal distribution with a mean of 0 m/s and a fixed standard deviation σ. No information about the environment is included in this model, and as a consequence, the filter permits the estimates to move along arbitrary paths. Although we know that in the system sensor nodes do not move, we still have to assign a movement model to enable particles to move closer to the actual location. Thus, our system model is simply $p(s_t|s_{t-1}) = N(0,\sigma)$, where N denotes the Normal distribution.

In summary, using the above system model, at every iteration each particle $s_{t-1}^{(i)}$ within the sample set X_v will be randomly moved from its original location based on the Normal distribution, resulting into particle $\tilde{s}_t^{(i)}$. The time complexity of this step is $O(N)$, where $N = |X|$ is the size of the sample set; the space complexity is also $O(N)$.

3.4 Measurement model

The measurement model is used to modify the initial particle $\tilde{s}_t^{(i)}$ to fit into the latest observation. Consequently, the measurement model depends on the observation data R_v as well as the reference distribution X_w. Here, we consider the following four types of observation data: (i) connectivity only, (ii) exact ranging, (iii) bounded ranging and (iv) AoA. Regardless of the observation data, the original particle $\tilde{s}_t^{(i)}$ is updated by comparing its location to the observation data using each particle in the distribution X_w as its reference. More weights are awarded to the particles that are more consistent with the observation. For every particle in X_v and X_w, the procedure AssignWeight is called to award the weight according to the observation data, R_v, as follows:

1. *Connectivity Only*: There are two types of connectivity only observations: positive and negative. We assume that all sensors are able to make connectivity observation at the very minimum. When R_v is a positive reading, it means nodes v and w are in the range of each other. Therefore, AssignWeight returns 1 if the distance between the two samples $d(\tilde{s}_t^{(i)}, s_t^{(j)}) \le d_{max}$, where d_{max} is the maximum range possible. Otherwise, AssignWeight returns 0. Conversely, when R_v is negative, it means that the two nodes are out of range; thus AssignWeight returns 0 if $d(\tilde{s}_t^{(i)}, s_t^{(j)}) \le d_{max}$, and 1 otherwise:

$$AssignWeight(\tilde{s}_t^{(i)}, s_t^{(j)}, R_v) = \begin{cases} 1, & R_v \text{ is } + \text{ and } d(\tilde{s}_t^{(i)}, s_t^{(j)}) \leq d_{max}, \\ & \text{or } R_v \text{ is negative and } d(\tilde{s}_t^{(i)}, s_t^{(j)}) > d_{max} \\ 0, & R_v \text{ is positive and } d(\tilde{s}_t^{(i)}, s_t^{(j)}) > d_{max}, \\ & \text{or } R_v \text{ is negative and } d(\tilde{s}_t^{(i)}, s_t^{(j)}) \leq d_{max} \end{cases}$$

2. *Exact Ranging*: When relatively reliable ranging data can be observed using methods like TOA, the ranging observation can be used directly to update the weights. Let the distance reading be $\hat{d}$. We compare the observed distance $\hat{d}$ with the sample distance $d(\tilde{s}_t^{(i)}, s_t^{(j)})$. In this case, we use the following evaluation equation

$$AssignWeight(\tilde{s}_t^{(i)}, s_t^{(j)}, R_v) = \begin{cases} 1 - (\frac{|d(\tilde{s}_t^{(i)}, s_t^{(j)}) - \hat{d}|}{d_{max}})^2, & |d(\tilde{s}_t^{(i)}, s_t^{(j)}) - \hat{d}| \leq d_{max} \\ 0, & \text{otherwise} \end{cases}$$

Comparing to the earlier case where only connectivity data is available, the above equation will award more weight to the particles more consistent with the observed distance $\hat{d}$. Also, the amount increases quadratically as the distance difference decreases. We note that more complex methods could be used; however our simulations indicated that this simple method works just as well as more complicated models.

3. *Bounded Ranging*: Ranging readings using methods such as RSSI are known to be unreliable due to multipath fading and scattering. In reality, the received signal strength would be usually weaker than what is expected based on the signal propagation model in the ideal environment. The distance estimate derived from such RSSI reading would be usually larger than the actual distance. In such cases, using the exact distance reading could adversely influence the location distribution. Thus, we use the bounded ranging observation, in which we merely regard the actual distance to be bounded by the observed distance. Let the observed distance reading be $\hat{d}$. For the bounded ranging observation, AssignWeight returns 1 when $d(\tilde{s}_t^{(i)}, s_t^{(j)}) \leq \hat{d}$, and 0 otherwise:

$$AssignWeight(\tilde{s}_t^{(i)}, s_t^{(j)}, R_v) = \begin{cases} 1, & d(\tilde{s}_t^{(i)}, s_t^{(j)}) \leq \hat{d} \\ 0, & \text{otherwise} \end{cases}$$

4. *AoA*: The AoA observation is similar to the exact ranging. Let $\hat{a}$ be to observed AoA reading, and $a(\tilde{s}_t^{(i)}, s_t^{(j)})$ be the AoA between the two particles. We use the following equation

$$AssignWeight(\tilde{s}_t^{(i)}, s_t^{(j)}, R_v) = \begin{cases} 1 - \left(\frac{|a(\tilde{s}_t^{(i)}, s_t^{(j)}) - \hat{a}|}{\pi}\right)^2, & |d(\tilde{s}_t^{(i)}, s_t^{(j)}) - \hat{d}| \leq d_{max} \\ 0, & \text{otherwise} \end{cases}$$

The above equation will only award weight to a particle if its calculated distance to the reference particle is in range. More weight is awarded to the particles that are more consistent with the AoA reading.

When the weights of all particles in X_v have been assigned, they are normalized so that the sum of all weights becomes one. Then, particles are re-sampled with the probability governed by the weight distribution. The new particle is generated using a Normal distribution centered at $\tilde{s}_t^{(i)}$ and a standard deviation being the average of X_v and X_w. The average is used here because our measurement model is influenced by both the original distribution X_v and the reference distribution X_w. As a common technique to avoid local minima, we also give a fixed p% (5% in our simulation) to randomly placed particles.

Overall, the time complexity of the measurement step is $O(N \cdot N)$, and the space complexity is $O(N)$. Algorithm 3 shows the FilterUpdate procedure including both the system model and measurement model. Notice that we actually skip the sample update steps if the variance of the reference distribution X_w is greater than the variance of the current sample distribution. This can be justified by the observation that it is unnecessary to update the current location samples when the reference samples are worse. Only when the reference samples are sounder than the current location samples should we run the update. This serves two purposes: (i) reducing the amount of unnecessary processing; and (ii) preventing the distribution to diverge when the neighbors adversely influence each other.

Algorithm 3 FilterUpdate(X_v, X_w, R_v)

```
if $var(X_v) < var(X_w)$ then
    return$X_v$
end if
$X_v \leftarrow$ sample set for node $v$ to be updated
$X_w \leftarrow$ reference sample set for node $w$ from which $R_v$ is observed
$R_v \leftarrow$ observation data
for all particle $(s_{t-1}^{(i)}, w_{t-1}^{(i)}) \in X_v$ do
    generate $\tilde{s}_t^{(i)} \leftarrow s_{t-1}^{(i)} \cdot N(0, \sigma)$
    $\tilde{w}_t^{(i)} \leftarrow 0$
    for all particle $(s_t^{(j)}, w_t^{(j)}) \in X_w$ do
        $\tilde{w}_t^{(i)} \leftarrow \tilde{w}_t^{(i)} +$ AssignWeight$(\tilde{s}_t^{(i)}, s_t^{(j)}, R_v)$
    end for
end for
```

Table c continued

```
Algorithm 3 FilterUpdate($X_v$, $X_w$, $R_v$)

Normalize weights and resample $s_t^{(i)}$ based on the weights
Randomly generate p% particles returnnew sample
    $X_v := \{(s_t^{(i)}, w_t^{(i)}) | i \in [1, N], w_t^{(i)} = 1/N\}$
```

3.5 Security concerns

Compared to pure distributed localization methods such as APS, our method can be regarded as "semi-centralized." However, unlike typical centralized algorithms which require the presence of all observation data, our method works more like an online algorithm, in which all computation is done at the beacon. While we will not provide detailed analysis and simulation due to page limit, it is easy to see that this model would make it easier to secure the localization process simply by securing the beacon itself. Let us assume that the beacon cannot be compromised, but the individual sensors can. A compromised sensor could attack the localization process in two ways: (i) reporting false observations to the beacon and (ii) interrupting the localization result. The report of false observations would affect the localization of the compromised sensor itself and more importantly those of the neighboring sensors one-hop away. However, given time the beacon would eventually move into the range of the neighbors and thus bypass the compromised sensor. Since the Monte Carlo sampling method is designed to work under uncertain observations, the false observations would eventually be filtered out as more direct observations are made. To help preventing the second type of attack, we require all messages sent by the beacon that contain the localization result to be encrypted with the receiver's unique key (or public key) and signed with the beacon's signature. This way, the compromised sensor will not be able to falsify the localization result. Although the decryption process can be expensive considering the limited capacity of the sensors, it needs to be performed one time only. A compromised sensor may still elect not to forward the localization result, however such attack can be countered by sending the localization results via multiple paths.

4 Simulation results

We have built a custom C++ discrete event simulation to evaluate the performance of our method. While our particle filter framework has no such restriction, we model all nodes in our simulation to have an identical transmission power. Thus, we can effectively control the network density (average degree) by varying the transmission range. When a node is located within the transmission range of another node, we assume that it is capable of receiving signal from the sender. We use a sensor network consisting of 49 nodes randomly deployed in a 1,000 m by 1,000 m square. A single mobile beacon flies through the deployment area. An epoch-based model controls the beacon movement. Starting from a random initial location, the beacon randomly chooses a velocity based on Normal distribution with a means of 20 m/s and a standard deviation of 20. The beacon proceeds to move at the chosen velocity for an epoch of time that is exponentially distributed with an expected value of 20 s. At the end of the epoch, the beacon chooses a new velocity and time. When the beacon reaches the boundary of the deployment area, it bounces back at the same incidence angle (much like a ping pong ball). While this model is much more realistic than a random motion model, we acknowledge that it still lacks many features of real beacon movement. For instance, airplanes or vehicles tend to make a smooth directional changes instead of a sudden turns.

Our simulation considers the following network parameters: the network connectivity is d, the signal to noise ratio is σ_{noise}, the number of particles used is N. For each new network scenario, we record the average estimation error and coverage, based on availability of four different types of location sensory data: (i) connectivity only, (ii) exact ranging, (iii) bounded ranging and (iv) AoA. We also consider the scenario where mixed types of sensory data co-exist in the same network. The final results shown in the figures are the average of 50 independent runs of different network topologies (calculated to provide 95% confidence with 5% relative error or better). We consider a sensor localized when its sample variance is less than a threshold T_{var}. Here we use $T_{var} = (d_{max}/2)^2$, that is, the standard deviation of the sample is less than half of the sensor range; this coverage criterion is rather arbitrary, as we can adjust to either a looser or a stricter threshold based on the actual application requirement. We also note, that the estimation error shown in the figures is the average of *all* nodes, including those not being localized.

4.1 Base scenario

First, our simulations concentrate on a basic scenario so that we have a benchmark to compare against other scenarios. For the basic scenario, we pick the average degree to be $c = 3.63$, the noise ratio to be $\sigma_{noise} = 0$ and the particle population to be $N = 200$. Figure 1 shows one such network. Networks of such low connectivity are highly likely to be disconnected. Previous research indicates that it can be very difficult to localize sparse networks using collaborative localization methods such as APS [4]. However, we will show that it is still possible to obtain reasonably good localization results using a mobile beacon.

Figure 2 shows the estimation error of every sensors in the same network of Fig. 1 after the beacon has traveled over the deployment area for 500 s; we assume that AoA sensory data is used. The lines originated from the nodes point to the estimated locations.

The average estimation error and the coverage for each of the four types of availability of sensory data are shown in Fig. 3. As the beacon spends more time over the deployment area, the estimation error decreases while the coverage increases, indicating that more nodes are being localized. After 200 s, the average estimation error is reduced below the set threshold. Since the basic scenario does not include any noise, sensory data based on exact ranging and AoA are accurate and thus we expect these results to be better. The best result is obtained using the AoA data. AoA data has an advantage over exact ranging because it takes only two (versus three) known locations to localize the sensor. Also note that the improvement of the estimation results slows down with time progressing. This is expected because of the (semi-) random beacon movement. As time advances it becomes less and less likely for the beacon to reach a previously unvisited area. Furthermore, the sensors that are the most difficult to localize reside at the boundary of the network, where they are less likely to be visited by the beacon.

4.2 One-hop neighbor observations

As stated in Algorithm 2, the localization algorithm does not only consider direct readings from the beacon to the

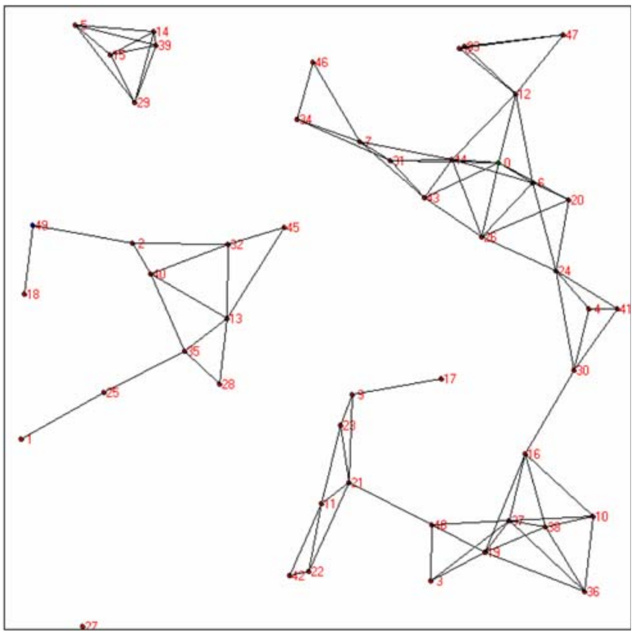


Fig. 1 Base scenario of a network of connectivity = 3.63. Two sensors are connected if a link exists in between

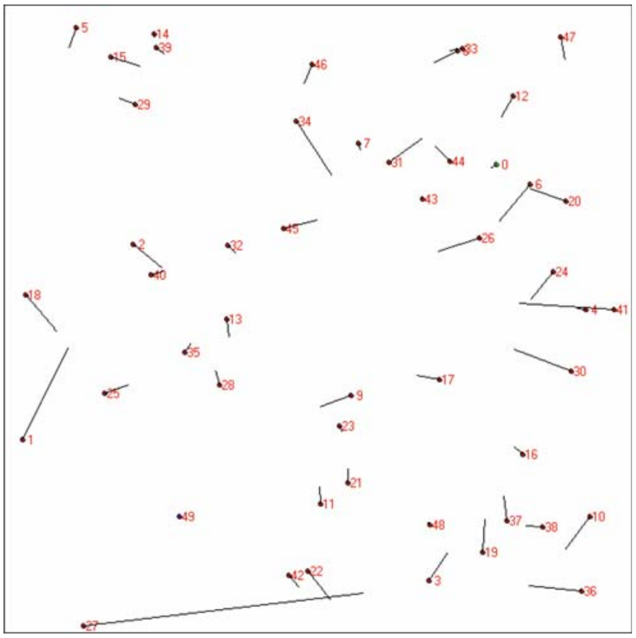
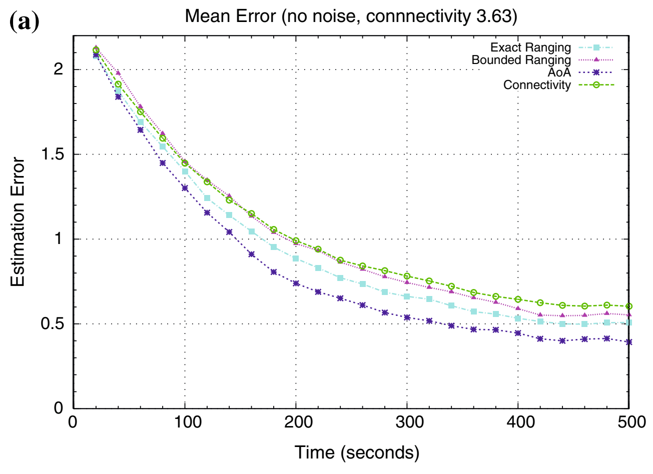


Fig. 2 Estimation error at the end of a 500 second run. The lines point to the estimated locations

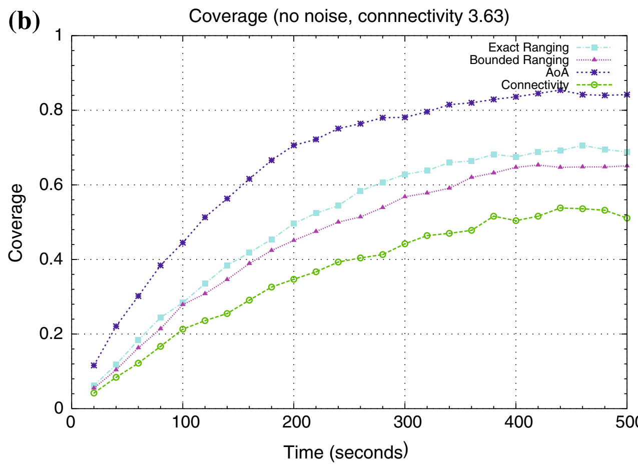
sensor, it also considers one-hop neighbor observations among sensors. Figure 4 shows results when one-hop neighbor observations are *not* used. In this case only the sensors that could directly overhear the beacon can be localized, which has been the traditional method of localization using a mobile beacon as proposed in [24]. However, as shown in Fig. 4, in such case the localization results are inferior compared to our basic case as the estimation error increases and the coverage decreases. In particular, the difference of the estimation error peaks between 200 s and 300 s, where up to 50% performance increase can be obtained by adding the one-hop neighbor observations. The advantage of using one-hop neighbor observations becomes less significant as time advances because the beacon will cover more and more previously not visited areas. Nevertheless, this experiment demonstrates the advantage of using one-hop neighbor observations especially when the beacon can only spend limited time over the deployment area.

4.3 Connectivity

Figure 5 contains the result when the average degree is increased from 3.63 to 5.47. We can observe that the change within the first 100 s is rather large; this can be credited to the fact that fewer sensors are localized during that time, which skews the result heavily. The change stabilizes as time advances and more sensors are localized. At the end of the 500 s simulation, we observe a decrease of estimation error by 25–50% and an increase of coverage

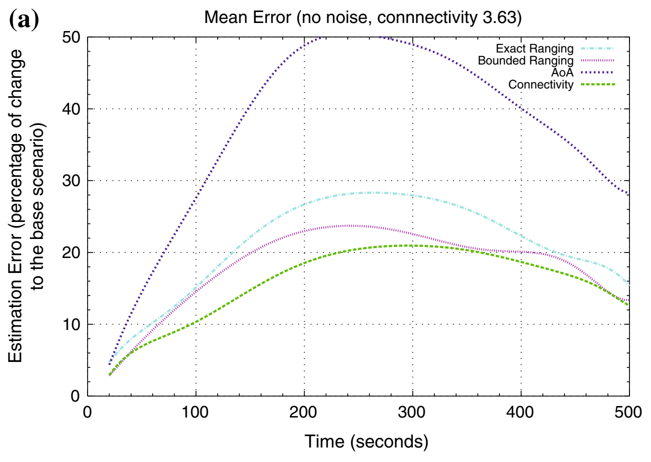


Estimation Error

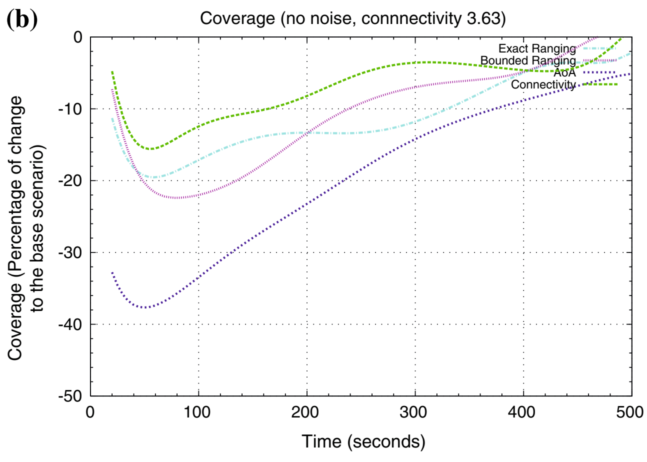


Coverage

Fig. 3 Result on the base scenario

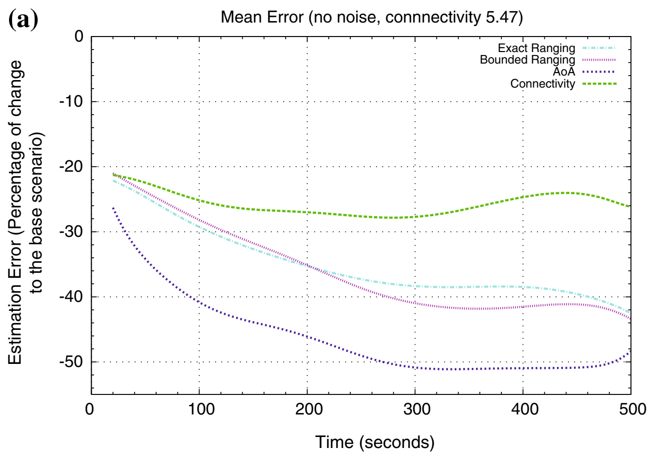


Estimation Error (Percentage of change to the base scenario)

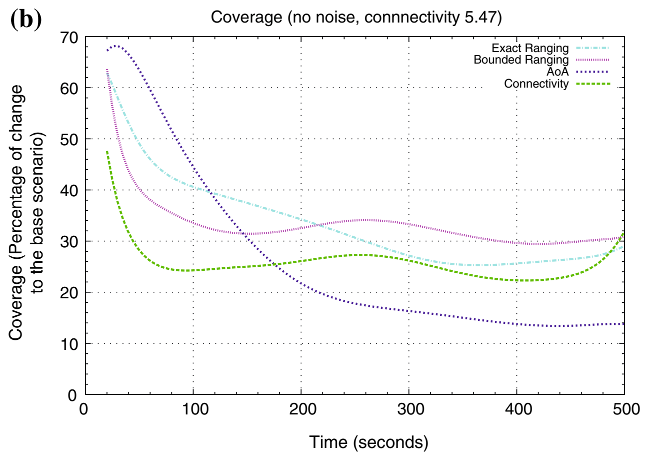


Coverage (Percentage of change to the base scenario)

Fig. 4 Result when 1-hop neighbor observations are *not* used



Estimation Error (Percentage of change to the base scenario)



Coverage (Percentage of change to the base scenario)

Fig. 5 Result of a denser network

by 15–30% when the connectivity increases to 5.47. This is to be expected as most localization methods work better on denser networks.

4.4 Noise

All of our previous experiments assumed an ideal scenario where location sensory data are not influenced by noise. Here we will consider a noisy environment with the random noise added to measurements. In the case of ranging, when the actual distance is *D*, the measured distance will be $D + D \cdot \textit{Uniform}(0, \sigma_{noise})$. In this case, we consider a situation where noise always *adds* to the actual distance (as described in the previous sections). The noise to AoA, on the other hand, is two-sided as it could either be *add*ed to or be *subtract*ed from the actual angle. Thus, if the actual angle is *A*, the measured angle will be $A + A \cdot \textit{Uniform}(-\textit{arcsin}(\sigma_{noise}), \textit{arcsin}(\sigma_{noise}))$ [8]. This makes the noise ratio comparable to ranging and AoA in that for the same noise ratio σ_{noise}, the amount of noise for AoA is about twice as much as that of ranging.

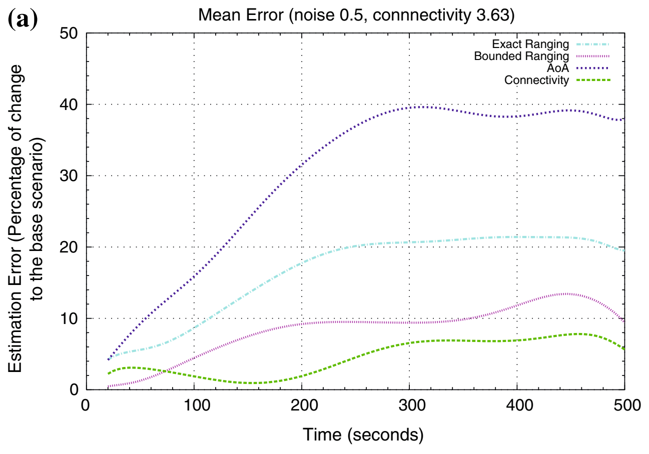
Figure 6 shows results when the noise ratio σ_{noise} is set to 0.5 (i.e., 50% noise). In all cases, the estimate's error increases and the coverage decreases when compared to the basic scenario. The increase is less significant for the bounded ranging and connectivity type scenarios because they rely less on the exact sensory readings. But exact ranging and AoA suffer up to 40% performance loss. As expected from the noise model, AoA degrades about twice as much as ranging.

4.5 Mixed sensory data

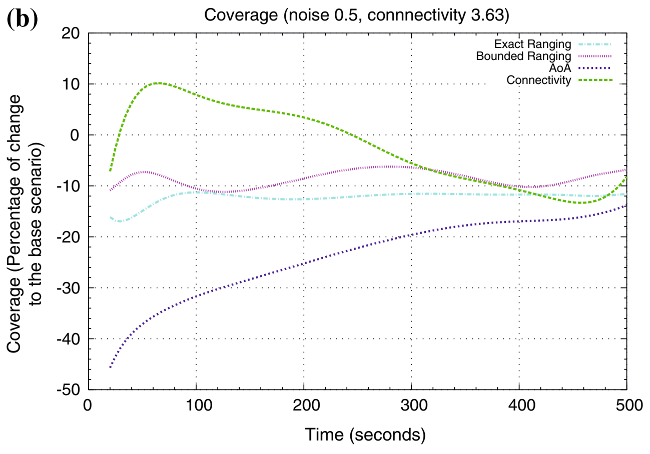
Since the same Monte Carlo localization framework can be applied to different sensory data, we also simulate the scenario where nodes are heterogeneous in their location sensory capabilities. We randomly assign a certain percentage of sensors to one of three location sensor types: exact ranging, AoA or connectivity-only; Fig.7 shows the results. It is interesting to note that adding AoA sensors helps to increase the localization performance. In particular, similar performance can be achieved using 1/3 of AoA and 1/3 of ranging when compared to the scenario where all sensors have ranging capacity.

5 Conclusions

In this paper, we proposed a localization method for wireless sensor networks using a probabilistic technique employing Monte Carlo sampling. We considered a scenario where a single mobile beacon passes through the deployment area. Location of nodes is computed on the beacon instead of the sensors. Possible sensor locations are represented by particle samples for each node, which are updated as the beacon passes through the network. The only tasks related to localization performed directly on the sensors are monitoring the sensory data from the neighbors and relaying localization results. This arrangement relieves the sensors of the heavy processing needed for accurate localization. At the same time, the use of sample sets allows us to maintain more complex localization distributions without sacrificing the resolution. Furthermore, since all computation is done at the beacon, the localization process can be more easily secured. Our method also allows sensors of different sensory capacities, such as ranging and AoA, to collaborate in the same network. Simulations show that our method is able to obtain less than 50% localization error and over 80% coverage on very sparse networks of degrees less than 4. Better result can be obtained by increasing the network connectivity.



Estimation Error (Percentage of change to the base scenario)



Coverage (Percentage of change to the base scenario)

Fig. 6 Result when noise ratio $\sigma_{noise} = 0.5$

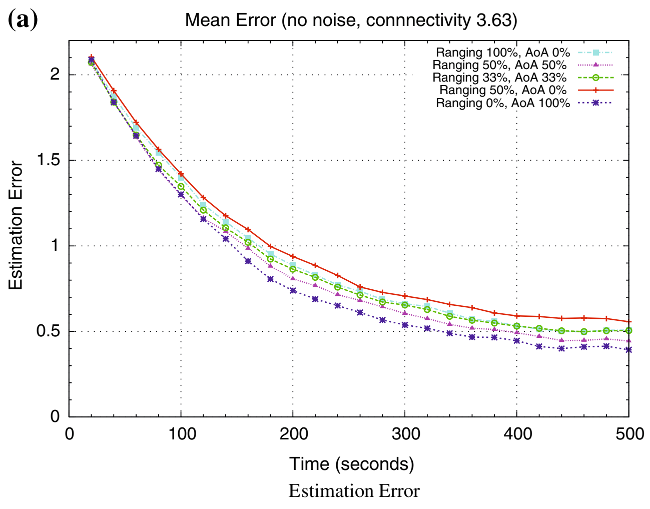


Fig. 7 Result of using mixed sensory data

Our current simulation assumes that the beacon moves based on a random movement profile. The movement profile can be improved such that the beacon would take an optimal path to increase the localization accuracy and minimize the localization time. In fact, the sample variances can be used as the guidance so that the beacon would move to the area where the overall variances are high. Furthermore, our method only deals with localization using a mobile beacon. Inter-sensor localization methods can be used after the mobile beacon exits the deployment area. Thus, the sensors that were unable to take advantage of the mobile beacon may still localize themselves by observing their neighbors.

References

1. Ahmed, A. A., Shi, H., & Shang, Y. (2005). SHARP: A new approach to relative localization in wireless sensor networks. In *Proceedings ICDCS Workshops*, pp. 892–898.
2. Biswas, P., & Ye, Y. (2004). Semidefinite programming for ad hoc wireless sensor network localization. In *Proceedings Third International Symposium on Information Processing in Sensor Networks*, pp. 46–54.
3. Bulusu, N., Heidemann, J., & Estrin, D. (2000). GPS-less low cost outdoor localization for very small devices. *IEEE Personal Communications Magazine, 7*(5), 28–34.
4. Chintalapudi, K., Govindan, R., Sukhatme, G., & Dhariwal, A. (2004). Ad-hoc localization using ranging and sectoring. In *Proceedings IEEE INFOCOM*, April, Hong Kong, China.
5. Galstyan, A., Krishnamachari, B., Lerman, K., & Pattem, S. (2004). Distributed online localization in sensor networks using a moving target. In *Proceedings IPSN*, pp. 61–70.
6. He, T., Huang, C., Blum, B. M., Stankovic, J. A., & Abdelzaher, T. F. (2003). Range-free localization schemes in large scale sensor networks. In *Proceedings ACM/IEEE MOBICOM*.
7. Hu, Y.-C., Perrig, A., & Johnson, D. (2003). Packet leashes: A defense against wormhole attacks in wireless ad hoc networks. In *Proceedings IEEE INFOCOM*, April.
8. Huang, R., & Záruba, G. V. (2007) Incorporating multiple sensory data for mobile adhoc networks localization. *IEEE Transactions on Mobile Computing, 6*(9), 1090–1104.
9. Jiang, X., & Camp, T. (2002). Review of geocasting protocols for a mobile ad hoc network. In *Proceedings Grace Hopper Celebration*.
10. Ko, Y., & Vaidya, N. H. (1998). Location-aided routing (LAR) in mobile ad hoc networks. In *Proceedings ACM/IEEE MOBICOM*, Dallas, TX.
11. Ko, Y., & Vaidya, N. (1999). Geocasting in mobile ad hoc betworks: location-based multicast algorithms. In *Proceedings IEEE Workshop on Mobile Computing Systems and Applications*.
12. Kuhn, F., Wattenhofer, R., Zhang, Y., & Zollinger, A. (2003). Geometric ad-hoc routing: Of theory and practice. In *Proceedings PODC*, pp. 63–72.
13. Langendoen, K., & Reijers, N. (2003). Distributed localization in wireless sensor networks: a quantitative comparison. *Computer Networks, 43*, 499–518.
14. Larsson, E. G. (2004). Cramer-Rao bound analysis of distributed positioning in sensor networks. *IEEE Signal Processing Letters, 11*(3), 334–337.
15. Liao, W.-H., Tseng, Y.-C., & Sheu, J.-P. (2001). GRID: A fully location-aware routing protocol for mobile ad hoc networks. *Telecommunication Systems, 18*(1), 37–60.
16. Liao, W.-H., Tseng, Y.-C., Lo, K.-L., & Sheu, J.-P. (2000). Geogrid: A geocasting protocol for mobile ad hoc networks based on grid. *Journal of Internet Technology, 1*(2), 23–32.
17. Mauve, M., Fuler, H., Widmer, J., & Lang, T. (2003). Position-based multicast routing for mobile ad-hoc networks, Technical report TR-03-004. Department of Computer Science, University of Mannheim.
18. Niculescu, D., & Nath, B. (2001). Ad hoc positioning system (APS). In *Proceedings IEEE GLOBECOM*, San Antonio.
19. Niculescu, D., & Nath, B. (2004). VOR base stations for indoor 802.11 positioning. In *Proceedings 10th Annual International Conference on Mobile Computing and Networking*.
20. Niculescu, D., & Nath, B. (2003). DV based positioning in ad hoc networks. *Telecommunication Systems, 22*(1–4), 267–280.
21. Niculescu, D., & Nath, B. (2003). Ad hoc positioning system (APS) using AoA. In *Proceedings IEEE INFOCOM*, San Francisco.
22. Peng, R., & Sichitiu, M. L. (2005). Robust, probabilistic, constraint-based localization for wireless sensor networks. In *Proceedings Second Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks*, September, Santa Clara, CA.

23. Savarese, C., Rabay, J., & Langendoen, K. (2002). Robust positioning algorithms for distributed ad-hoc wireless sensor networks. In *Proceedings USENIX Technical Annual Conference*, June, Monterey, CA.
24. Sichitiu, M. L., & Ramadurai, V. (2004). Localization of wireless sensor networks with a mobile beacon. In *Proceedings First IEEE Conference on Mobile Ad-hoc and Sensor Systems*, Fort Lauderdale, FL.
25. Ssu, K.-F., Ou, C.-H., & Jiau, H. C. (2005). Localization with mobile anchor points in wireless sensor networks. *IEEE Transactions on Vehicular Technology, 54*(3), 1187–1197.
26. Vural, S., & Ekici, E. (2004). Wave addressing for dense sensor networks. In *Proceedings IEEE Wokshop on Sensor and Actor Network Protocols and Applications*, pp. 56–66.
27. Xu, Y., Heidemann, J., & Estrin, D. (2001). Geography-informed energy conservation for ad hoc routing. In *Proceedings ACM/IEEE MOBICOM*, pp. 70–84.

Author Biographies

Rui Huang has received the Ph.D. in Computer Science from The University of Texas at Arlington in 2006, and the B.S. degree in Computer Science from Texas Christian University in 1998. He has held a number of positions in software industry. His research interests include wireless networks, sensor networks, network protocols and algorithms. He is a member of the IEEE and its Communications Society.

Gergely V. Záruba is an Assistant Professor of Computer Science and Engineering at The University of Texas at Arlington (CSE@UTA). He has received the Ph.D. degree in Computer Science from The University of Texas at Dallas in 2001, and the M.S. degree in Computer Engineering from the Technical University of Budapest, Department of Telecommunications and Telematics, in 1997. Dr. Záruba's research interests include wireless networks, algorithms, and protocols, performance evaluation, current wireless and assistive technologies. He has served on many organizing and technical program committees for leading conferences and has guest edited journals. He is a member of the IEEE and its Communications Society.