Localization in Wireless Sensor Networks

Masoomeh Rudafshani and Suprakash Datta
Department of Computer Science and Engineering
York University
Toronto, ON, Canada
masoomeh@cse.yorku.ca, datta@cse.yorku.ca

ABSTRACT

A fundamental problem in wireless sensor networks is localization – the determination of the geographical locations of sensors. Most existing localization algorithms were designed to work well either in networks of static sensors or networks in which all sensors are mobile. In this paper, we propose two localization algorithms, MSL and MSL*, that work well when any number of sensors are static or mobile. MSL and MSL* are range-free algorithms – they do not require that sensors are equipped with hardware to measure signal strengths, angles of arrival of signals or distances to other sensors. We present simulation results to demonstrate that MSL and MSL* outperform existing algorithms in terms of localization error in very different mobility conditions. MSL* outperforms MSL in most scenarios, but incurs a higher communication cost. MSL outperforms MSL* when there is significant irregularity in the radio range. We also point out some problems with a well known lower bound for the error in any range-free localization algorithm in static sensor networks.

Categories and Subject Descriptors: C.2 [Network Architecture and Design]: Wireless communication

General Terms: Algorithms, Performance.

Keywords: Sensor networks, localization, mobile, Monte Carlo sampling, lower bounds.

1. INTRODUCTION

Sensor networks are composed of large numbers of sensors that are equipped with a processor, memory, wireless communication capabilities, sensing capabilities and a power source (battery) on-board. While in most existing sensor networks, sensors are static, some modern applications involve sensors that are mobile. For example, in habitat monitoring applications like ZebraNet [15] sensors are attached to zebras and collect information about their behavior and migration patterns [27]. In other applications, sensors are deployed on cellular phones to measure reception quality [27].

IPSN'07, April 25-27, 2007, Cambridge, Massachusetts, USA.

A fundamental problem in designing sensor network is localization – determining the location of sensors. Location information is used to detect and record events, or to route packets using geometric-aware routing, *e.g.*, [16]. Manual configuration of locations is not feasible for large-scale networks or networks where sensors may move. Providing each sensor with localization hardware (e.g., GPS [13]) is expensive in terms of cost and energy consumption. A more reasonable solution to the localization problem is to allow some nodes (called *seeds*) to have their location information at all times, and allow other nodes to infer their locations by exchanging information with seeds.

Localization algorithms can be divided into two categories: *range-based* and *range-free*. In range-based algorithms, nodes estimate their distance to seeds using some specialized hardware. These measurements are used in methods like triangulation or trilateration [1], which are based on the idea that a node location is uniquely specified when at least the coordinates of three reference points are available for a node. Although the use of range measurements results in a fine-grained localization scheme, range-based algorithms require the sensors contain hardware to make range measurements.

Range-free algorithms do not use radio signal strengths, angle of arrival of signals or distance measurements and do not need any special hardware. Range-free algorithms require that each node knows
(a) which nodes are within radio range
(b) their location estimates.
(c) the (ideal) radio range of sensors.
No other information is used for localization. Thus, range-free techniques are more cost-effective because they do not require sensors to be equipped with any special hardware, but use less information than range-based algorithms.

1.1 Existing work

Much of the existing research focuses on static sensor networks. Relatively less is known about localization in mobile sensor networks, and very few algorithms work in situations where the sensors may be static or mobile. We survey some relevant papers in this section, and refer the user to the surveys by Bachrach and Taylor [1] and by Savvides *et al.* [25] for more comprehensive literature reviews.

Range-based localization: Various techniques have been proposed for measuring the distances in range-based algorithms. Bahl *et al.* [2] and Bischoff *et al.* [4] used received signal strength (RSS) to estimate distances. Ward *et al.* [29] used *Time of Arrival* of signals, Priyantha *et al.* [23] and Savvides *et al.* [24] have used *Time Difference of Arrival* of signals to estimate distances, and Niculescu and Nath [22]

have used *Angle of Arrival* of signals to estimate distances. Recently Havinga *et al.* [7] proposed a novel range-based algorithm based on the Monte Carlo approach.

Range-free localization: He *et al.* [12] proposed a range-free algorithm called APIT in which all possible triangles of the seeds are formed. The location of a node is the center of intersection region of all triangles. Nagpal *et al.* [20] proposed the Gradient algorithm which uses a method similar to distance vector routing to allow the nodes to find the number of hops to all the seeds. Seeds also estimate the average distance per hop using a range-free technique and send this estimate to the nodes. Nodes then use multilateration to find their locations. Niculescu and Nath [21] proposed DV-Hop which is similar to Gradient, but uses a different method for estimating the average distance per hop. Sextant [11] proposed a distributed algorithm that uses Bèzier curves for estimating the possible locations of the nodes. The nodes estimate their locations by using the location information of their neighbors. Recently, a new approach to range-free localization called the radiointerferometric approach was proposed in [17, 18].

Localization in mobile sensor networks: None of the above mentioned algorithms consider the features of a mobile sensor network. Tilak *et al.* [27] studied how often a localization algorithm should be applied in a mobile sensor network. This involves a tradeoff between energy and accuracy – localizing too often wastes energy and localizing too infrequently results in having stale locations at most of the time. Bergamo and Mazzimi [3] proposed a range-based algorithm for mobile sensor networks which uses only two static seeds that are located at a specific location in the network and their radio ranges cover the whole network. Recently, we proposed a range-free localization algorithm [6] that works in both static and mobile networks. The main problem associated with the algorithm in [6] is that it cannot estimate locations well when the radio ranges of sensors are not perfect circles.

Our work is inspired by Hu and Evans [14], who proposed a range-free algorithm, *MCL* (*M*onte *C*arlo *L*ocalization), that only works in mobile sensor networks. In MCL, sensors move in an ad hoc manner and the only information available to a node is its maximum speed, v_{max}. MCL is based on the Monte Carlo method [8] which has been used for robot localization [10] as well. In MCL, possible locations of a node are represented with a set of sample locations, which are updated (as new observations arrive) using the Monte Carlo approach. In addition, when seeds move they provide the coordinate of a new location. MCL works well in mobile sensor networks as long as the speed of movement is not very low. The drawbacks of MCL are

(1) it needs a high density of seeds, and

(2) the sampling technique it uses to generate probable locations is very slow and computation-intensive (the implementation of MCL uses a relaxation parameter that reduces, but does not eliminate, this problem).

1.2 Our contributions

We propose (in Section 2) two fully range-free algorithms called MSL (*M*obile and *S*tatic sensor network *L*ocalization), and MSL* that work well when some or all nodes are static or mobile. Both algorithms can handle heterogeneity in radio transmission range, however, MSL* has more communication and computation cost than MSL.

In Section 3, using simulations experiments we show that both MSL and MSL* outperform existing algorithms (Gradient [20] and MCL [14]) in terms of localization error in a wide variety of parameter values. MSL and MSL* exhibit graceful degradation in performance with decreasing seed density. The location estimates of MSL and MSL* converge faster than MCL and the sampling procedure they use is faster than that of MCL.

Finally, we describe some problems with the lower bound for the localization error in any static range-free localization algorithm proved by Hu and Evans [14].

Our results can be viewed in several ways.

(1) We improve and generalize MCL [14] and demonstrate how to improve localization accuracy by using the location estimates of all neighbors (not just seeds). Of course, this makes our algorithms use more communication and computation at the sensor nodes. Thus our algorithms are more appropriate for larger sensors that can support the extra communication.

(2) Our results illustrate the tradeoff between accuracy and computation and communication complexity.

2. OUR MODEL AND ALGORITHMS

We consider a network in which a small fraction of sensors (called *seeds*) are equipped with hardware, *e.g.*, GPS, that allow them to know their location at all times; all sensors are otherwise identical. Our algorithms can handle heterogeneity in radio range, but we assume for ease of exposition that each sensor has the same ideal radio range, r.

We model irregularity in radio range by assuming that the radio range of a sensor follows a normal distribution with mean r and standard deviation σ. A greater σ value results in more irregularity. We assume that each sensor knows the value of r. The simulator uses σ to randomly determine for each packet whether the sender and the receiver are within radio range. We chose this distribution over the uniform distribution used in [14] because we wanted to allow greater variation than allowed by the uniform distribution, albeit with small probabilities. Our model is a very coarse approximation of real radio ranges. Our experiments suggest that the results obtained with our Gaussian model and the uniform model in [14] do not differ significantly for our performance metrics. We acknowledge that this is a crude model of real radio ranges, which are affected by noise, fading, antenna directionality, the presence of obstacles and many other factors. Accurate modeling of radio ranges is beyond the scope of this paper. The interested reader is referred to [28] for a more detailed model of radio ranges.

Nodes are initially deployed randomly over the network area. The *first hop neighbors* of sensor p are those sensors that can communicate with it directly. The *second hop neighbors* of sensor p are those sensors that can communicate with the first-hop neighbors of p directly, but are not themselves first-hop neighbors of p. We assume that time is discretized. The algorithms do not assume very tightly synchronized clocks. Each node and seed is capable of moving a distance v in a time step in any direction where $0 \leq v \leq v_{max}$. The nodes know v_{max}, but they do not know the value of v or the direction of movement in any time step. MSL and MSL* are based on the Monte Carlo method which we outline in the next section and describe how it is relevent to localization problem.

2.1 The Monte Carlo Method

In applications where the state of a system has to be estimated from some observations, the system can be formulated using a Bayesian model in which the posterior distribution of the state of the system is only dependent on the current observations and current state of the system [8]. In dynamic systems, observations arrive sequentially and so it is required to update the posterior distribution of the system upon arrival of new observations. In the Monte Carlo method, the distribution of the state of the system is represented with a set of samples. The set of samples is updated when new observations arrive.

Different Monte Carlo approaches have been proposed in literature. In this work, we use the particle filtering approach [8, 9]. Particle filtering is used in robotics for estimating the location of a robot [10]. This technique is fully distributed and easy to implement. The key idea in particle filtering is to represent data distribution of the system by a set of N weighted samples:

$$p(S_t|O_{0\ldots t}) \approx \{s_t^{(i)}, w_t^{(i)}\}_{i=1,\ldots,N}$$

where $p(S_t|O_{0\ldots t})$ is the distribution representing the state of the system at time t, $s_t^{(i)}$ is a sample of the state of the system at time t, and $w_t^{(i)}$'s are non-negative numerical weights that sum to one. The number of samples maintained by the system is an important parameter: a minimum number of samples should be available so that the set of the samples converges to the posterior distribution of the system (see Doucet *et al.* [8] and Tanner [26] for details). The steps of the Monte Carlo method are as follows:

1. Initialization: N samples are chosen from the initial distribution of the system, $p(S_0)$.
2. Sampling: N samples, $\tilde{s}_t^i$ for $i = 1, \ldots, N$, are drawn from the distribution $p(S_t|S_{t-1})$ where $p(S_t|S_{t-1})$ is the transition equation or motion model. Then the weight of each sample is computed using the observations. Finally, the weights are normalized, $w^{(i)} = \eta\tilde{w}_t^{(i)}$, where η is the normalizing factor.
3. Re-sampling: N samples are chosen with replacement from the current sample set according to their weights.

In both our algorithms, each node maintains a set of weighted samples denoting its possible locations. The location of a node is estimated as the weighted mean of its samples. Each node updates its samples in every time step (after each movement) using the Monte Carlo method.

In the remainder of the paper, $d(a, b)$ denotes the distance between locations a, b and r denotes the ideal radio range. We describe algorithm MSL* first.

2.2 Algorithm MSL*

MSL* maintains a set of probable locations (samples) for each node. These samples are assigned weights that estimate their quality. Intuitively, the weight of a sample is an approximation of the likelihood that it represents the true location of the node, *given* the estimated locations of neighbors. The steps of algorithm are as follows.

Initialization: In the first step, nodes have no information about their location. The first set of samples for each node is chosen randomly from the whole sensor field. In this step, the nodes only use the seeds within their neighborhood for weighting the samples, since the non-seed neighbors have no useful information.

Sampling: In this step, nodes generate new samples using the following transition equation:

$$p(S_t|S_{t-1}) = \begin{cases} \frac{1}{\pi(v_{max}+\alpha)^2} & \text{if } d(S_t, S_{t-1}) \leq v_{max} \\ 0 & \text{if } d(S_t, S_{t-1}) > v_{max} \end{cases}$$

where v_{max} is the maximum speed of a node and $d(S_t, S_{t-1})$ denotes the distance between the locations of a sample at time t and $t-1$. Every time step, a new sample is generated from each current sample by randomly choosing a point within a circle centered at the current location of the sample and radius $v_{max} + \alpha$. The parameter α is needed for MSL* to work when no sensors move. There is a tradeoff involving α: large values of α increase the amount of uncertainty in the location estimates, especially for mobile nodes, where we know the new sample should be in a circle with radius v_{max}. If α is too small it cannot provide enough variability in choosing new samples when the speed of sensors is low, or zero. The value of α used in this paper, $\alpha = 0.1r$, was determined empirically.

After choosing a sample, its weight is determined using the neighborhood information as follows. The weight of a sample s chosen for node p, $w_s(p)$, is computed as follows: corresponding to each neighbor q of node p, we find a partial-weight for sample s, $w'_s(q)$. The weight of sample s is the product of the partial-weights obtained corresponding to each neighbor of node p. That is,

$$w_s(p) = \Pi_{q=1}^{k} w'_s(q) \tag{1}$$

where k is the number of first-and second-hop neighbors of node p, and q denotes a neighbor of node p. The partial weights $w'_s(q)$ corresponding to the first-and second-hop neighbors q are computed as follows.

The partial weight of sample s corresponding to a first-hop seed neighbor q is

$$w'_s(q) = \begin{cases} 1 & \text{if } d(s, q) \leq r \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

The partial weight corresponding to a second-hop seed neighbor q is

$$w'_s(q) = \begin{cases} 1 & \text{if } r \leq d(s, q) \leq 2r \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

The partial weight of sample s corresponding to a first-hop node neighbor q is computed using the weights $w(q_i)$ of samples q_i of node q as follows:

$$w'_s(q) = \sum_{q_i} w(q_i), \text{ where } d(s, q_i) \leq r + v_{max}. \tag{4}$$

Similarly, for second-hop node neighbors q, $w'_s(q)$ is computed as follows:

$$w'_s(q) = \sum_{q_i} w(q_i), \text{ where } r - v_{max} \leq d(s, q_i) \leq 2r + v_{max}. \tag{5}$$

Sample s is kept if $w_s(p)$ is greater than a threshold value, β. Parameter β is a real number in the interval $[0, 1]$ and its value depends on the number of neighbors of a node. Therefore, different nodes have different β values. The parameter β should be chosen so that as the number of neighbors of a node increases, its value decreases. The reason is that $w_s(p)$ is the product of the numbers which are at most one (the

If (node not localized or number of samples are zero)
  If (node has first-hop or second-hop neighbors)
    find N samples with weights greater than β
    Normalize the weights of the samples
  Else
    closeness = ∞
    keep the last set of samples
Else
  Sample (α) (Sampling step with parameter α)
  If no sample found
    closeness = ∞
    keep the last set of samples
  Normalize weights
Resample the sample set (Re-sampling step)
Send locations and closeness to first- and second-hop neighbors.

Figure 1: MSL* algorithm in every node

partial weights corresponding to the nodes are usually less than one and the partial weights corresponding to the seeds are one or zero). In this paper, we use $\beta = (0.1)^{\mathrm{t}}$, where t is the number of first-and second-hop neighbors of a node.

After computing $w_s(p)$ using Eqn. 1, the weights are normalized to make sure that the sum of the weights of the samples is equal to one. Therefore, if N samples are chosen for node p, the weight of the i-th sample is normalized as:

$$\frac{w_i(p)}{\sum_{j=1}^{N} w_j(p)}.$$

Resampling: The purpose of resampling step is to gradually remove samples with lower weights and keep those with higher weights. In this step, each node computes a new sample set from its current sample set. Each sample in the current sample set is included in the new sample set with a probability proportional to its weight. Since the number of samples is kept fixed, in this step, a sample with a small weight has a lower chance of being selected, and a sample with a higher weight has a greater chance of being selected, and so many duplicates of that sample are likely to exist in the new sample set.

The pseudo-code of the MSL* algorithm is given in figure 1. Note that the algorithm has the same basic structure as the distance vector algorithm used for propagating routing information.

As mentioned earlier, each node uses the location estimates of its neighbors to weight its samples. However, a node uses only the information of those neighbors that have a *more accurate* estimate of their locations. This has the dual advantage of reducing communications costs and using only reliable location estimates.

The quality of a location estimate is measured using a parameter that we call *closeness*. The closeness value for node p with N samples is computed as follows:

$$closeness_p = \frac{\Sigma_{i=1}^{N} w_i \sqrt{(x_i - x)^2 + (y_i - y)^2}}{N}$$

where N is the number of samples of node p, (x_i, y_i) denotes the coordinate of the i-th sample ($i = 1 \ldots N$), w_i denotes the weight of the i-th sample, and (x, y) is the current location estimate of node p. The closeness for a seed is always zero and the closeness for a node is a number greater than zero. Intuitively, lower closeness values indicate higher accuracy in the location estimate of a node. In Section 3, using simulation experiments, we show that the closeness is a good measure of accuracy of location estimate of a node.

At the beginning of the MSL* algorithm, the closeness values for the seeds are zero and the closeness values for the nodes are ∞. Therefore, in the first time step only the seeds provide location information for their first-and second-hop neighbors. As the algorithm proceeds, nodes update their location estimates as well as their closeness. Each node sends to its neighbors its location (or estimates) and its closeness.

Whenever a node moves to a location at which it has no neighbors, it does not receive any new location information. In this case, the previous sample set is used to estimate the location of the node. In the next time step, the node is re-localized, *i.e.*, the current sample set is re-initialized.

In MSL*, each node uses the location information of all its first-and second-hop neighbors and this results in more communication compared to MCL. We will quantify the communication costs of both these algorithms in Sec. 3.3. We now describe an algorithm MSL that has significantly lower communication costs than MSL*.

2.3 Algorithm MSL

The communication-intensive part of MSL* is the transfer of samples between the nodes. As shown in Eqns. 4 and 5, each node uses the samples of (some of) its neighbors to weight its samples. In MSL, we assign a weight to each *node*. Every node uses the weights of its neighbors (rather than weights of samples of neighbors) to weight its samples. After the weights are computed, MSL computes a *a single location estimate* (the weighted mean of samples) and a closeness value. Each node broadcasts to its neighbors this estimate and its closeness value (but not its samples). Thus, the communication cost drops significantly.

In MSL, we would like the weight assigned to each node to depend on the quality of its location estimate. Therefore, we define the weight of a node as a function of its closeness value as follows:

$$w_q = b^{-closeness_q}.$$

The performance of MSL was not sensitive to the choice of $b \in [2, 9]$; we use $b = 7$ in this paper. Like MSL*, in MSL a node uses the locations of only those neighbors that have lower closeness values.

In MSL, the weights of seed neighbors are computed just as in MSL* (equations 2, 3). The weights of first-hop non-seed neighbors (equation 4 for MSL*) are computed as follows:

$$w'_s(q) = \begin{cases} w_q & \text{if } d(s,q) \leq r + v_{max} + v_{extra} \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

Since nodes use less information in MSL than MSL* (a single location estimate of neighbors rather than a set of weighted samples) so, we need to allow for some extra uncertainty in the location estimates of the nodes for MSL – this is done using parameter v_{extra}. MSL was not sensitive to the choice of $v_{extra} \in [0.2r, 0.5r]$; we use $v_{extra} = 0.3r$ in this paper. The weights of second-hop non-seed neighbors (equation 5

for MSL*) are computed as follows:

$$w'_s(q) = \begin{cases} w_q & \text{if } r - v_{max} - v_{extra} \leq d(s,q) \\ & \leq 2r + v_{max} + v_{extra} \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

In section 3, we compare the performance of MSL* and MSL algorithm and discuss their pros and cons.

2.4 Differences with MCL

Our algorithms use Monte Carlo method, like MCL [14]. However, we improve on MCL and generalize it in several ways. First, we modify the sampling procedure in MCL to allow our algorithm to work in static networks. This also allows our algorithms to outperform MCL even when our algorithm uses only location information from seed neighbors.

Second, in both MSL and MSL* each node uses information from only those neighbors that have better location estimates (measured using the *closeness* parameter) than it. This modification yields improved performance in networks where nodes have zero or low speeds, and low seed densities, and results in faster convergence of the algorithms.

Third, we modify the sampling procedure and allow samples to have weights greater than a threshold value, β. This results in faster convergence of the localization error, faster execution time, and hence better estimation of the locations in mobile networks.

For MCL to work, the sensors need to move with at least a predetermined minimum speed. MCL does not work when sensors move at low speeds, and does not work in static networks. We will show that MSL* estimates locations with high accuracy when the sensors are static, in the case they move at low speeds, and when sensors move at high speeds.

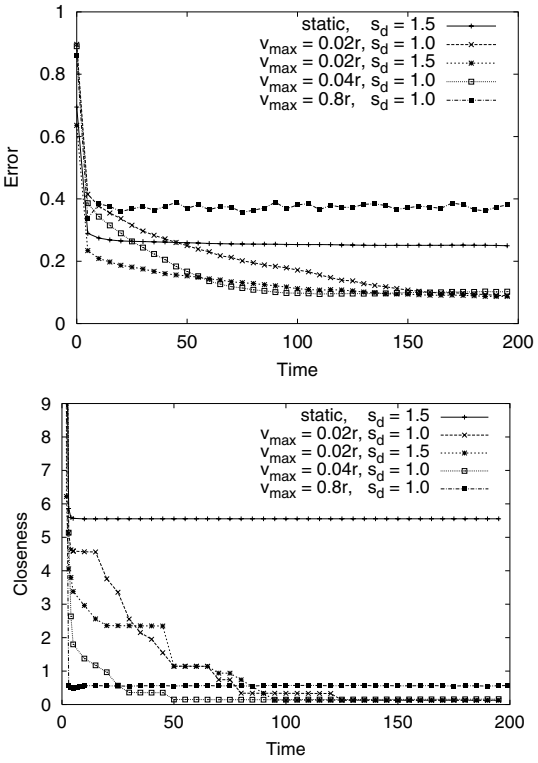

Figure 2: Convergence of the error and closeness values of the MSL* algorithm under different mobility conditions.

3. PERFORMANCE EVALUATION

We received a simulator for MCL and Gradient from Hu and Evans [14] written in Java. We implemented MSL and MSL* in this simulator and performed our simulations using it. We define next the performance metrics used to compare localization algorithms and enumerate simulation parameters that are used in the experiments.

3.1 System Model and Parameters

The key metric for evaluating a localization algorithm is the accuracy of the location estimates or *localization error*. This is computed as follows:

$$\text{Error} = \frac{1}{n} \sum_{i=1}^{n} ||e_i - l_i|| \quad (8)$$

where n is the number of sensors, l_i denotes the real location of the i-th node and e_i denotes the location estimate of the i-th sensor where $i = 1, \ldots, n$. For the seeds, the estimated location is the same as the real location, since we assume that the seeds know exactly where they are. The errors shown in the simulation results are in terms of the radio range, *i.e.*, the errors shown are computed by dividing the error in equation 8 by the radio range of sensors. Each data point in a graph is computed by averaging the result of 50 simulation experiments. The mobility of all nodes are held exactly to the same values in each simulation.

Most parameter settings for our simulations are those used in [14]. Our results were obtained using sensors randomly distributed in a 500 units × 500 units square field. Our algorithms outperform others by a bigger margin for irregular-shaped (e.g. a 'U'-shape) sensor fields but those results are omitted due to space constraints.

We set ideal radio range $r = 100$ in our experiments. The *node density*, n_d, is the average number of sensors, including both nodes and seeds, in the neighborhood of a sensor. *Seed density*, s_d, is the average number of seeds in the neighborhood of a sensor. We note that n_d, s_d should really be called node and seed *degrees* respectively; however, for notational consistency with [14], we will call them densities in this paper. In our simulations, we set $n_d = 10$ and $s_d = 1$ unless otherwise specified.

Movement of sensors is implemented using a modified version of random waypoint mobility model [5] used by Hu and Evans [14]. This model prevents nodes from pausing at waypoints and thus prevents decay in average speeds that occurs under the random waypoint model [14, 30].

3.2 Simulation Results

In this section, we compare the MSL, MSL*, Gradient, and MCL algorithms under various network configurations.

Convergence of MSL*: Figure 2 shows the convergence of MSL* algorithm under different mobility conditions. The figure shows that in all situations the error in location estimates goes down until it reaches a stable state in which the error fluctuates slightly about a constant value.

In the static case, the error stabilizes very fast because location estimates do not change much after the informa-

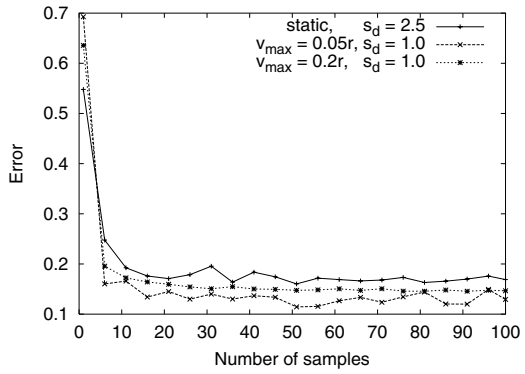

Figure 3: Effects of the number of samples in MSL*.

tion from seeds reaches all the nodes. In contrast, mobile nodes can get more information from the environment by visiting more sensors and gradually refine their locations. So, movement can provide the opportunity for having more observations and getting more accurate location estimates. However, when the speed of movement is high, nodes benefit less from the previous location estimates and only the information from the last time step affects the localization.

Figure 2 shows the convergence of the error and the closeness values. The figure shows that as time proceeds, both error in location estimates and the closeness values decrease. This validates our assumption in Section 2 that the closeness value is a reasonable measure of the quality/reliability of location estimates.

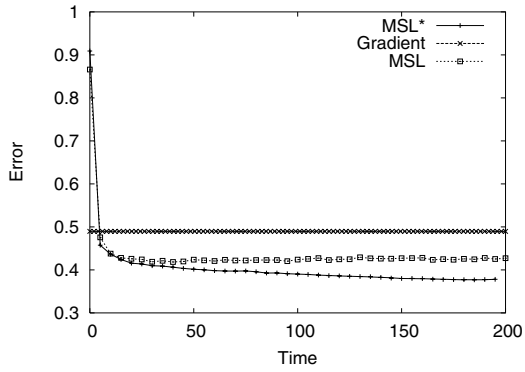

Figure 4: Accuracy comparison, static nodes.

The change in error and closeness in MSL is qualitatively similar to that in MSL*, but the errors are a little higher in MSL. We omit the graphs for MSL due to space limitations.

Determining the number of samples: In a Monte Carlo method, for the set of samples to converge to the distribution of samples, a minimum number of samples should be available [9]. Figure 3 shows the result of simulation with different number of samples and different network configurations. Based on this graph and in keeping with [14], we use 50 samples in our simulations.

Accuracy Comparison of Different Algorithms: The graphs in Figure 4, 5 show the accuracy comparison for MSL*, Gradient, MCL, and MSL under various mobility

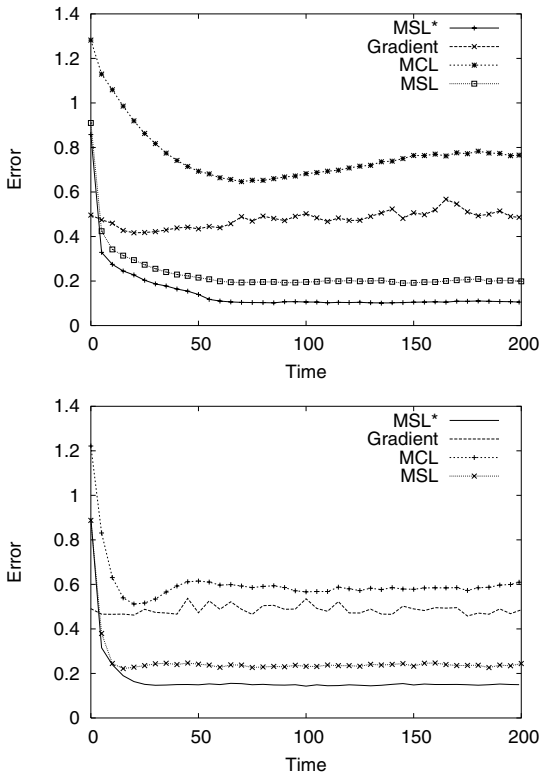

Figure 5: Accuracy for $v_{max} = 0.05r$ and $v_{max} = 0.2r$.

speeds (recall that MCL does not work when nodes are static). It can be seen from these figures that MSL and MSL* outperform both MCL and Gradient.

Effect of the Speed of Mobile Sensors: Figure 6 shows how the localization error varies with changing the speed of sensors. In MCL, MSL*, and MSL small values of the speed reduces the error until it reaches a minimum value. Further increase in the speed reduces the accuracy and increases the error. The reason for this behavior is that adding some movement in the network can increase the chance of visiting more sensors and getting more information from the environment. However, at high speeds, past location information is no longer useful and the error increases.

In the implementation of the Gradient algorithm, it is assumed that the sensors have enough time to get location information from all the seeds. This assumption gives unfair advantage to Gradient and so, movement has no effect on its performance.

As shown in Figure 6, MSL* and MSL estimate locations more accurately than MCL and Gradient at lower speeds. The performance of MSL deteriorates at high speeds because in MSL the nodes use estimated locations (and not the samples) of their neighbors for localization. So the localization error is higher when nodes move at higher speeds.

The graphs show that most algorithms have good performance when $v_{max} = 0.2r$. So, we use this value for simulations involving variation of other parameters.

Effect of Node and Seed Density: In this experiment we first kept number of seeds in the network constant and varied the number of non-seed nodes. As shown in the first graph in Figure 7, a higher number of nodes results in better

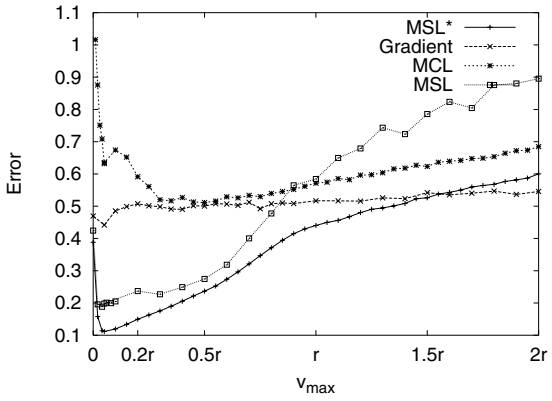

Figure 6: Effect of the speed of mobile sensors.

localization in all algorithms. However, in MSL* and MSL, as the number of the nodes goes up, the error in location estimates goes down. The reason is that when there is a higher number of nodes, each node has more sensors in its first-and second-hop neighborhood and gets more location information, so the localization improves. The improvement is more pronounced for static sensor networks which cannot benefit from visiting new neighbors due to movement.

In MCL increasing the nodes provides communication with more second hop seed neighbors and so localization improves [14]. In Gradient because all nodes cooperate in spreading the information and computing the average distance per hop, increasing number of nodes has more effect on the algorithm.

Next, we varied the number of seeds, keeping the node density fixed. The second graph in Figure 7 shows that increasing the seed density reduces the error in all algorithms. As long as a minimum number of seeds are available, the Gradient, MSL and MSL* algorithms work well and increasing the number of seeds has less effect on the algorithms. In MCL, each node only uses the seeds within its neighborhood to estimate its location. Therefore, its accuracy is much more dependent on the number of seeds.

Our simulations, using the simulator used in [14] show that Gradient indeed does benefit from increased seed density, as is intuitively expected. This is seen in the first graph in Figure 7. So, contrary to [14], we find that Gradient outperforms MCL at some seed densities.

Effects of Irregularity in Radio Range: The irregularity in radio range is measured using the parameter σ described in section 2. Figure 8 shows that as irregularity in the radio range increases, error in estimation goes higher in all algorithms. However, the performance of MSL degrades more gracefully than other algorithms. The intuitive explanation of this is that we find plausible is that when the data is noisy it is better to process averages (which smooth out fluctuations) than actual samples.

3.3 Communication and Computation Cost

MSL and MSL* use information from non-seed nodes in localization (unlike MCL), and so have greater communication cost than MCL. However, in MSL each node only broadcasts its estimated location whereas in MSL* all samples of a node are broadcast to neighbors.

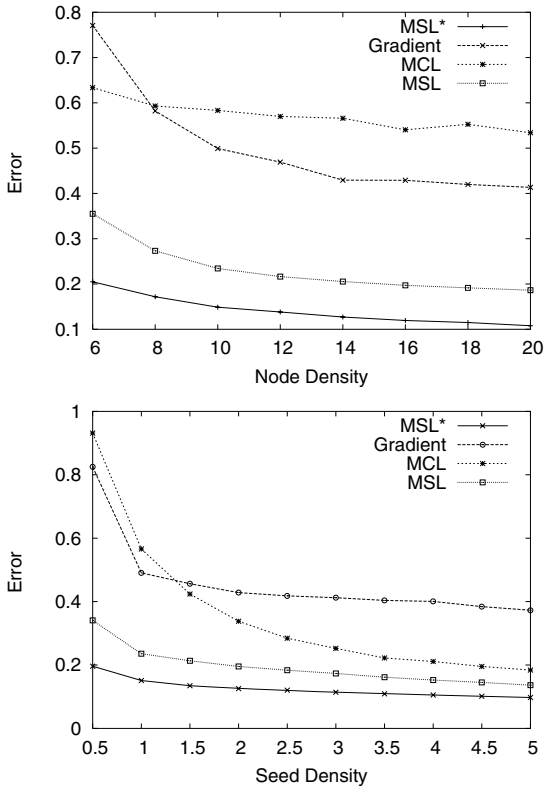

Figure 7: Effects of the node and seed density.

In MSL*, the communication cost is proportional to $n*s+m$, where n is the number of nodes in the network, m is the number of seeds, and s is the number of samples maintained by each node. In MSL, the cost is proportional to $n+m$. In MCL, communication cost is proportional to m.

In terms of computation cost, MSL and MSL* are less expensive than MCL, because the sampling in MCL takes much longer. Also, the faster convergence speed of MSL and MSL* implies that it requires less computation than MCL.

It is worth pointing out that if we restrict the MSL* algorithm so that each node uses *only* the seeds within its neighborhood, the computation and communication costs drop drastically, since nodes are no longer involved in the localization of other nodes. The performance of this restricted version of MSL* is shown in Figure 9. The figure shows that MSL* can estimate locations more accurately than MCL while having less communication and computation cost than MCL. This is due to the improved sampling algorithm of MSL*.

4. SOME NOTES ABOUT LOWER BOUNDS

Nagpal *et al.* [20] have claimed that $r\pi/4n_d$ is a lower bound for the error in any range-free localization algorithm in static sensor networks where the nodes only use the connectivity information of the seeds within their first-hop neighborhood. Hu and Evans [14] extended their idea to the algorithms in which the nodes use information of their second-hop and their first-hop neighbors in static networks. In this section, we outline the lower bound proposed by Nagpal *et al.* [20], and then discuss problems associated with it.

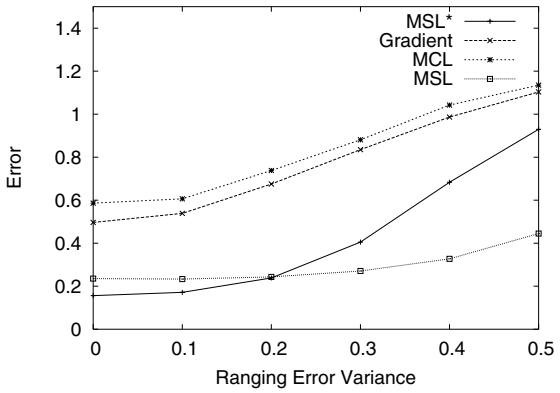

Figure 8: Effects of irregularity in the radio ranges.

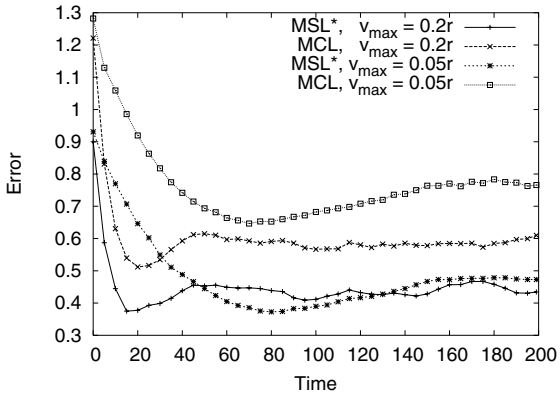

Figure 9: Comparison of MCL and MSL* when MSL* uses only seeds for localization.

4.1 Outline of the Lower Bound Proof

In the network model Nagpal *et al.* [20] have used for proving the lower bound, the x and y coordinates of the sensors are chosen randomly and independently from a value between the boundaries of the network area. They used the result of work by Mendenhall *et al.* [19] which states that in this scenario, the probability that there are k sensors in a given area a follows a Poisson distribution:

$$\Pr(\text{ k sensors in area a}) = \frac{(\rho a)^k}{k!} e^{-\rho a}$$

where ρ is equal to n/A in which A is the total area of the network region and n is the total number of sensors in area A. From the above formula, they concluded that the expected number of sensors in area a would be ρa. A sensor can hear all sensors within circle with radius r around it. Therefore, the node density n_d is $\rho\pi r^2$.

Let the continuous random variable Z represent the maximum distance sensor p can be moved without changing its neighborhood. Then, the expected value $E[Z]$ is a lower bound on the localization error of nodes. They computed the distribution of Z as follows.

The probability that $Z < z$ is equal to the probability that there is at least one sensor in the shaded area, $A(z)$, shown in Figure 10. So, we have:

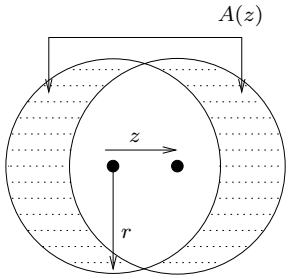

Figure 10: A sensor can move distance z without changing the connectivity if there is no sensor in the shaded area.

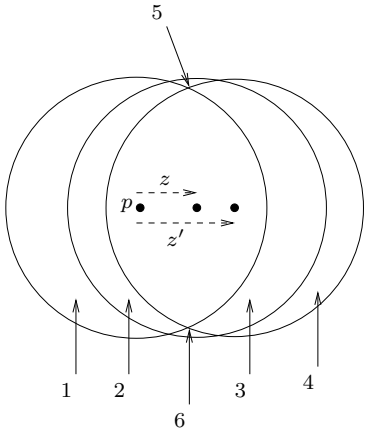

Figure 11: if there is any sensor in areas 5 or 6 and no one in areas 1, 2, 3, and 4, sensor p can move distance z' without changing the connectivity, however, it cannot move distance z without changing the connectivity, while $z' > z$.

$$F(z) = \Pr(Z \le z) = 1 - e^{-\rho A(z)}$$

They approximated the area $A(z)$ in Figure 10 as $A(z) \approx 4rz$ by assuming that z is small compared to r for reasonable densities of sensors. Therefore, $E[Z]$ can be computed as:

$$E(Z) = \int_0^\infty z\dot{F}(z)dz \quad (9)$$

$$= \int_0^\infty \rho 4rze^{-\rho 4rz}dz \quad (10)$$

$$= \frac{1}{\rho 4r} \quad (11)$$

replacing n_d by $\rho\pi r^2$, we have

$$E(Z) = \frac{\pi r}{4n_d} \quad (12)$$

4.2 Problems with the lower bound proof

There is a problem associated with the analysis mentioned in the previous section. Recall the distribution found for Z. In Figure 11, assume that there is no sensor in areas 1,2,3, and 4, but there is at least one sensor in areas 5 or

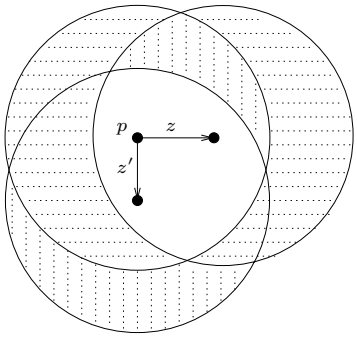

Figure 12: A sensor can move distance z in one direction but cannot move distance z' in another direction while $z' < z$.

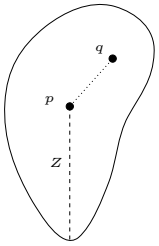

Figure 13: Region with the same connectivity as p.

6. Therefore, sensor p can move distance z' without changing the connectivity, but it cannot move distance z without changing the connectivity, while $z < z'$. This problem is easy to fix – we can compute the expression for the area of the shaded region ($A(z)$) in Figure 10 plus the area of the blackened region at the top and bottom of the circles. For $z < 2r$, this area is given by

$$A(z) = \pi r^2 + 2rz - 2r^2 \cos^{-1}\left(\frac{z}{2r}\right) + z\sqrt{r^2 - z^2/4}.$$

The proof is a straightforward trigonometric calculation and is omitted for lack of space.

Notice now that for a rigorous lower bound, we need to show that $A(z) \leq 4rz$ and $\dot{A}(z) \geq 4r$ in equation 10. These would of course be true if $A(z) = 4rz$ but this is not true. It can be verified that the bound $A(z) \leq 4rz$ holds, but $\dot{A}(z) \geq 4r$ is *not true*. Thus the lower bound is not rigorous.

Next, we discuss another aspect of the lower bound proof and possible ways to improve it. One problem with the distribution of Z is that the *direction* is not considered. However, direction is an important issue – in one direction a sensor may be able to move a longer distance than the other directions. For example, as shown in Figure 12, sensor p can move distance z towards the right if there is no sensor in the horizontally shaded area, but it cannot move distance z' downwards, if there is any sensor in the vertically shaded region, while $z' < z$. Note that if there is any sensor in vertically shaded region, it does not affect the movement with distance z towards the right.

This problem affects the expression for finding the expected distance a sensor can move without changing its neighborhood. Figure 13 shows the region with the same connectivity as sensor p. Let q be a random point within this region and Y be a random variable denoting the distance between p and q. The lower bound for the average error of any range-free localization algorithm would be $E[Y]$ which is equal to the average distance a sensor can move without changing the connectivity. The distance Z shown in this figure is the Z defined by Nagpal *et al.* [20].

We investigated $E[Y]$ by computing its value numerically for various node densities. We generated a large number of random sensors and a large number of random points in the network area. The average distance between sensors and points with the same connectivity information would be $E[Y]$. The results for various node densities are shown in Figure 14. Also, the lower bound expression from [20] has been shown in this figure. The graph shows that the bound is particularly loose at lower node densities.

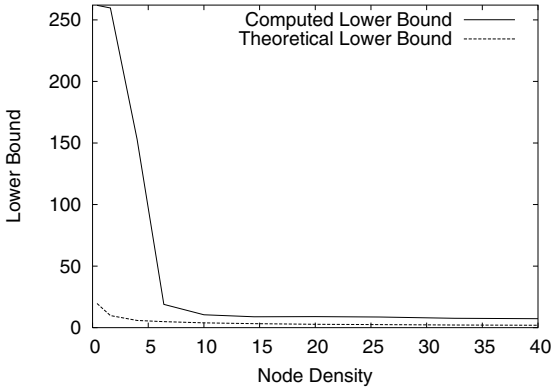

Figure 14: Comparison of Lower bound from [20] and computed lower bound, $E[Y]$.

Note that although Y has the same density function along each fixed direction, we cannot simply fix the direction to compute $E[Y]$. Therefore, we cannot simply compute $E[Y]$ along a fixed direction. A better lower bound would use the absence of sensors in all directions.

5. CONCLUSIONS

We proposed two algorithms (MSL, MSL*) for node localization in wireless sensor networks, that work in both static and mobile sensor networks. Both outperform two existing state-of-the art algorithms in terms of localization accuracy and exhibit low dependency on the number of seeds. In situations where degree of irregularity in radio range is high, MSL outperforms all other algorithms.

Acknowledgment

The authors thank NSERC for financial support. They also gratefully acknowledgethe code sent by the Hu and Evans [14].

6. REFERENCES

[1] J. Bachrach and C. Taylor. *Handbook of Sensor Networks*, chapter Localization in Sensor Networks. Wiley, 2005.

[2] P. Bahl and V. Padmanabhan. RADAR: An in-building RF-based user location and tracking system. In *Proceedings of INFOCOM*, pages 775–784, 2000.

[3] P. Bergamo and G. Mazzimi. Localization in sensor networks with fading and mobility. In *The 13th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, pages 750–754, 2002.

[4] U. Bischoff, M. Strohbach, M. Hazas, and G. Kortuem. Constraint-based distance estimation in ad-hoc wireless sensor networks. In *Proceedings of the Third European Workshop on Wireless Sensor Networks (EWSN)*, pages 54–68, 2006.

[5] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless Communications and Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, 2(5):483–502, 2002.

[6] S. Datta, C. Klinowski, M. Rudafshani, and S. Khaleque. Distributed localization in static and mobile sensor networks. In *IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 69–76, 2006.

[7] B. Dil, S. Dulman, and P. J. M. Havinga. Range-based localization in mobile sensor networks. In *Proccedings of the Third European Workshop on Wireless Sensor Networks (EWSN)*, pages 164–179, 2006.

[8] A. Doucet, N. Freitas, and N. Gordon. *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, 2001.

[9] A. Doucet, S. Godsill, and C. Andrieu. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and Computing*, 10(3):197–208, 2000.

[10] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte carlo localization: Efficient position estimation for mobile robots. In *AAAI 1999*, pages 343–349, 1999.

[11] S. Guha, R. Murty, and E. Sirer. Sextant: a unified node and event localization framework using non-convex constraints. In *ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pages 205–216, 2005.

[12] T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. Abdelzaher. Range-free localization schemes for large scale sensor networks. In *Proceedings of the 9th annual international conference on Mobile computing and networking (MobiCom)*, pages 81–95, 2003.

[13] B. Hofmann-WeUenhof, H. Lichtenegger, and J. Collins. *Global Positioning System, Theory and Practice*. Springer-Verlag, 1993.

[14] L. Hu and D. Evans. Localization for mobile sensor networks. In *Proceedings of the 10th annual international conference on Mobile computing and networking (MobiCom)*, pages 45–57, 2004.

[15] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with zebranet. In *Proceedings of the 10th international conference on Architectural support for programming languages and operating systems (ASPLOS)*, pages 96–107, 2002.

[16] B. Karp and H. T. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 243–254, 2000.

[17] B. Kusý, A. Lédeczi, M. Maróti, and L. Meertens. Node density independent localization. In *Proceedings of the fifth international conference on Information processing in sensor networks (IPSN)*, pages 441–448, 2006.

[18] M. Maróti, P. Völgyesi, S. Dóra, B. Kusý, A. Nádas, A. Lédeczi, G. Balogh, and K. Molnár. Radio interferometric geolocation. In *Proceedings of the 3rd international conference on Embedded networked sensor systems (SenSys)*, pages 1–12, 2005.

[19] W. Mendenhall, D. Wackerly, and R. Scheaffer. *Mathematical Statistics with Applications*. PWS-Kent Publishing Company, 1989.

[20] R. Nagpal, H. Shrobe, and 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)*, pages 333–348, 2003.

[21] D. Niculescu and B. Nath. Ad hoc positioning system (APS). In *Proceedings of GLOBECOM*, pages 2926–293, 2001.

[22] D. Niculescu and B. Nath. Ad hoc positioning system (APS) using AoA. In *Proceedings of INFOCOM*, pages 1734–1743, 2003.

[23] N. Priyantha, A. Chakraborty, and H. Balakrishnan. The cricket location-support system. In *Proceedings of the Sixth Annual ACM International Conference on Mobile Computing and Networking (MOBICOM)*, pages 32–43, 2000.

[24] A. Savvides, C.-C. Han, and 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)*, pages 166–179, 2001.

[25] A. Savvides, M. Srivastava, L. Girod, and D. Estrin. *Wireless sensor networks*, chapter Localization in sensor networks, pages 327–349. Kluwer Academic Publishers, Norwell, MA, USA, 2004.

[26] M. Tanner. *Tools for Statistical Inference, Theory and Practice, 3rd edition*. Springer-Verlag, 1996.

[27] S. Tilak, V. Kolar, N. B. Abu-Ghazaleh, and K. D. Kang. Dynamic localization control for mobile sensor networks. In *24th IEEE International Performance, Computing, and Communications Conference (IPCCC)*, pages 587–592, 2005.

[28] M. Torrent-Moreno, F. Schmidt-Eisenlohr, H. Fussler, and H. Hartenstein. Effects of a realistic channel model on packet forwarding in vehicular ad hoc networks. In *Wireless Communications and Networking Conference (WCNC)*, volume 1, pages 385–391, 2006.

[29] A. Ward, A. Jones, and A. Hopper. A new location technique for the active office. *IEEE Personal Communications*, 4(5):42–47, 1997.

[30] J. Yoon, M. Liu, and B. Noble. Sound mobility models. In *MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 205–216, 2003.