Localization for Mobile Sensor Networks

Lingxuan Hu   David Evans
Department of Computer Science
University of Virginia
Charlottesville, VA
{lingxuan, evans}@cs.virginia.edu

ABSTRACT

Many sensor network applications require location awareness, but it is often too expensive to include a GPS receiver in a sensor network node. Hence, localization schemes for sensor networks typically use a small number of seed nodes that know their location and protocols whereby other nodes estimate their location from the messages they receive. Several such localization techniques have been proposed, but none of them consider mobile nodes and seeds. Although mobility would appear to make localization more difficult, in this paper we introduce the sequential Monte Carlo Localization method and argue that it can exploit mobility to improve the accuracy and precision of localization. Our approach does not require additional hardware on the nodes and works even when the movement of seeds and nodes is uncontrollable. We analyze the properties of our technique and report experimental results from simulations. Our scheme outperforms the best known static localization schemes under a wide range of conditions.

Categories and Subject Descriptors

C.2.1 [**Computer Systems Organization**]: Network Architecture and Design – *Distributed networks*, *Wireless communication*; G.3 [**Mathematics of Computing**]: Probability and Statistics – *Probabilistic algorithms (including Monte Carlo)*.

General Terms

Algorithms, Design, Experimentation.

Keywords

Localization, sensor networks, mobility, Monte Carlo Localization.

1. INTRODUCTION

Location awareness is important for wireless sensor networks since many applications such as environment monitoring, vehicle tracking and mapping depend on knowing the locations of sensor nodes. In addition, location-based routing protocols can save significant energy by eliminating the need for route discovery [25, 26, 32] and improve caching behavior for applications where requests may be location dependent [28]. Security can also been enhanced by location awareness (for example, preventing wormhole attacks [22, 24]). However, putting GPS receivers in every node or manually configuring locations is not cost effective for most sensor network applications.

Recently some localization techniques have been proposed to allow nodes to estimate their locations using information transmitted by a set of seed nodes that know their own locations (for example, because they have GPS receivers). These techniques are described in Section 2. They all suffer from one or both of these problems:

1. *Dependence on special hardware.* Techniques that depend on measuring ranging information from signal strength [2], time of arrival [43], time difference of arrival [40] or angle of arrival [36] require hardware that is typically not available on sensor nodes. Adding the required hardware increases the cost and size of the nodes.
2. *Requirement for particular network topologies.* Most techniques require seed nodes to be numerous and evenly distributed so they can cover the whole network. But prior deployment of seeds is not possible in many sensor network applications (for example, sensor nodes dropped from plane over a hostile territory). Hop count based techniques [35, 34] avoid the need for a large number of seeds, but instead require dense and uniform node distribution.

We are interested in performing localization in a more general network environment where no special hardware for ranging is available, the prior deployment of seed nodes is unknown, the seed density is low, the node distribution is irregular, and where nodes and seeds can move uncontrollably. Although mobility makes other localization techniques increasingly less accurate, our technique takes

advantage of mobility to improve accuracy and reduce the number of seeds required.

We consider a network composed of nodes with unknown locations and seeds that know their locations. We are interested in three kinds of scenarios:

1. *Nodes are static, seeds are moving.* For example, a military application where nodes are dropped from a plane onto land, and transmitters attached to soldiers or animals in the area are used as moving seeds. Each node's location estimate should become more accurate as time passes and it receives information from more seeds.

2. *Nodes are moving, seeds are static.* One example would be nodes floating in currents along a river and seeds at fixed locations on the river banks. A more concrete, but less worldly, example is NASA's Mars Tumbleweed project [1]. It proposes a low cost way to explore large areas on Mars by having rovers with sensors that are blown over the surface by the wind, with minimal or no control over their movement. Of course, GPS does not work on Mars, but it may be possible to establish fixed landmark seeds or positioning from orbiters [31]. For these scenarios, each node's location estimate will fluctuate around its current actual location: as time passes, old location information becomes inaccurate since the node has moved, but as new seed information is received the location estimate is revised.

3. *Both nodes and seeds are moving.* This is the most general situation. It applies to any application where the nodes and seeds are both deployed in an ad hoc way, and move either because of the environment they are in (wind, currents, etc.) or because they have actuators for motion.

Next we provide background on previous localization work. In Section 3, we introduce our localization algorithm. The algorithm is analyzed in Section 4. Section 5 reports on simulated experiments and compares our results with other localization techniques.

2. BACKGROUND

Extensive research has been done on localization for wireless networks. A general survey is found in [20]. Here, we provide only a brief survey focusing only on localization techniques suitable for ad hoc sensor networks. The approaches taken to achieve localization in sensor networks differ in their assumptions about the network deployment and the hardware's capabilities.

Centralized localization techniques depend on sensor nodes transmitting data to a central location, where computation is performed to determine the location of each node. Doherty, Pister and Ghaoui developed a centralized technique using convex optimization to estimate positions based only on connectivity constraints given some nodes with known positions [11]. MDS-MAP [41] improves on these results by using a multidimensional scaling approach, but still requires centralized computation. Requiring central computation would be infeasible for mobile applications because of the high communication costs and inherent delay, hence we focus on distributed localization techniques.

Distributed localization methods do not require centralized computation, and rely on each node determining its location with only limited communication with nearby nodes. These methods can be classified as range-based and range-free. Range-based techniques use distance estimates or angle estimates in location calculations, while a range-free solution depends only on the contents of received messages.

Range-based approaches have exploited time of arrival [43], received signal strength [2, 37] time difference of arrival of two different signals (TDOA) [40], and angle of arrival (AOA) [36]. Though they can reach fine resolution, either the required hardware is expensive (ultrasound device for TDOA, antenna arrays for AOA) or the results depend on other unrealistic assumptions about signal propagation (for example, the actual received signal strengths of radio signals can vary when the surrounding environment changes).

Because of the hardware limitations of sensor devices, range-free localization algorithms are a cost effective alternative to more expensive range-based approaches [19]. There are two main types of range-free localization algorithms that have been proposed for sensor networks: local techniques that rely on a high density of seeds so that every node can hear several seeds, and hop counting techniques that rely on flooding a network.

Local Techniques. In the Centroid method [6], each node estimates its location by calculating the center of the locations of all seeds it hears. If seeds are well positioned, location error can be reduced [7], but this is not possible in ad hoc deployments. The APIT method [19] isolates the environment into triangular regions between beaconing nodes, and uses a grid algorithm to calculate the maximum area in which a node will likely reside. Since APIT typically assumes a larger radio range for seed nodes and hence has high seed density, it is not reasonable to compare it to our technique directly.

Hop-Counting Techniques. To provide localization in networks where seed density is low, hop-counting techniques propagate location announcements throughout the network. DV-HOP [35] uses a technique based on distance vector routing. Each node maintains a counter denoting the minimum number of hops to each seed, and

updates that counter based on messages received. Seed location announcements propagate through the network. When a node receives a new seed announcement, if its hop count is lower than the stored hop count for that seed, the recipient updates it hop count to the new value and retransmits the announcement with an incremented hop count value. The Amorphous localization algorithm [34] uses a similar approach. The coordinates of seeds are flooded throughout the network so each node can maintain a hop-count to that seed. Nodes calculate their position based on the received seed locations and corresponding hop count.

None of these schemes target the case where nodes or seeds can move. They can be adapted for mobile networks by refreshing location estimates frequently, but are not designed with any consideration for how mobility can be exploited to achieve localization. The only work we are aware of that considers localization with mobile nodes is Bergamo and Mazzini's [3]. They considered mobility of nodes only and assumed a network with two seeds at fixed locations that can transmit across the entire network, and nodes that are able to measure signal strength accurately. Instead of using mobility to improve localization, they studied how mobility makes localization more difficult and found that errors increased with increasing node speed.

On the other hand, the mobile localization problem has been extensively studied in robotics. Robotics localization work usually assumes a prior or previously learned map, and tries to determine the robot's position based on its motion and sensor data. If both the motion and the measurement model can be described using a Gaussian density, and the initial state is also a Gaussian, then a classical Kalman filter [33] can be used. Grid based Markov localization [4, 5] has been proposed to deal with multimodal and non-Gaussian densities. However, the grid representation can impose a significant memory and computational burden, especially if one is interested in high resolution.

Ladd et al. used a robotics localization approach to achieve accurate localization for wireless networks by using learned variation in RF signal strengths received using standard Ethernet cards [29, 30]. Their approach performs well for indoor localization in fixed environments, but assumes fixed seed locations and requires a learning phase, so is not well suited to mobile sensor network applications.

Our work adapts the Monte Carlo localization (MCL) method [10, 42] developed for use in robotics localization for use in mobile sensor network applications. MCL is a particle filter combined with probabilistic models of robot perception and motion. It outperforms other proposed localization algorithms in both accuracy and computational efficiency. The key idea of MCL is to represent the posterior distribution of possible locations using a set of weighted samples. Each step is divided into a prediction phase and an update phase. In the prediction phase, the robot makes a movement and the uncertainty of its position increases. In the update phase, new measurements (such as observations of new landmarks) are incorporated to filter and update data. The process repeats and the robot continually updates its predicted location.

Like our work, MCL applies the Sequential Monte Carlo method [18] to achieve localization. However, there are substantial differences between robot localization and node localization for sensor networks. While robot localization locates a robot in a predefined map, localization in sensor networks works in a free space or unmapped terrain. Second, a robot has relatively good control and probabilistic knowledge of its movement in a predefined map. A sensor node typically has little or no control of its mobility, and is unaware of its speed and direction. Third, a robot can obtain precise ranging information from landmarks, but a sensor node can only learn that it is within radio range. Finally, in robot localization, the individual measurements are integrated multiplicatively, assuming conditional independence between them, and the weights of samples need to be normalized after updating. In our algorithm, due to the constraints in computing and memory power, we adopt a filtering approach in which each measurement can be considered independently, and the weight of each sample is either 0 or 1.

3. LOCALIZATION PROTOCOL

The constraints in sensor nodes and ranging precision make localization for mobile sensor nodes a more difficult problem than robot localization. On the other hand, scale can be used to our advantage. The many nodes in a sensor network can cooperate to share location information.

We assume time is divided into discrete time units. Since a node may move away from its previous location, it needs to re-localize in each time unit. We are interested in obtaining the probabilistic distribution of a node's possible locations. As a node moves in the network, prior location information will become increasingly inaccurate. On the other hand, there are new observations from seed nodes that are able to filter impossible locations. The posterior distribution of a node's possible locations after movement and observation is not easy to determine. Except for a few special cases including linear Gaussian state space models (Kalman filter [33]), it is impossible to evaluate the distribution analytically [12].

The Sequential Monte Carlo (SMC) method [18] provides simulation-based solutions to estimate the posterior distribution of nonlinear discrete time dynamic models. The key idea of SMC is to represent the posterior distribution by

Initialization: Initially the node has no knowledge of its location N is a constant that denotes the number of samples to maintain

$\qquad L_0 = \{$ set of N random locations in the deployment area $\}$

Step: Compute a new possible location set L_t based on L_{t-1}, the possible location set from the previous time step, and the new observations, o_t.

$L_t = \{\ \}$
while (size $(L_t) < N$) **do**

$\qquad R = \{\, l_t^i \mid l_t^i$ is selected from $p(l_t \mid l_{t-1}^i)$, $l_{t-1}^i \in L_{t-1}$ for all $1 \leqslant i \leqslant N\,\}$ $\qquad$ **Prediction** (3.2)

$\qquad R_{filtered} = \{\, l_t^i \mid l_t^i$ where $l_t^i \in R$ and $p(o_t \mid l_t^i) > 0\,\}$ $\qquad\qquad\qquad$ **Filtering** (3.3)

$\qquad L_t = $ choose $(L_t \cup R_{filtered}, N)$

Figure 1. Location estimation algorithm.

a set of m weighted samples, and to update them recursively in time using the importance sampling method [16]. Since the unconditional variance of the importance weights will increase [27], re-sampling techniques [39] are used to eliminate trajectories with small normalized importance weights. SMC has been successfully applied in target tracking [17], robot localization [10] and computer vision [23]. We provide a brief introduction below. A more detailed introduction can be found in [13], and an overview and discussion of SMC's properties can be found in [12].

3.1 Location Estimation Algorithm

The mobile localization problem can be stated in a state space form as follows. Let t be the discrete time, l_t denote the position distribution of the node at time t, and o_t denote the observations from seed nodes received between time t-1 and time t. A transition equation $p(l_t \mid l_{t-1})$ describes the prediction of node's current position based on previous position, and an observation equation $p(l_t \mid o_t)$ describes the likelihood of the node being at the location l_t given the observations. We are interested in estimating recursively in time the filtering distribution $p(l_t \mid o_0, o_1, \ldots, o_t)$. A set of N samples L_t is used to represent the distribution l_t, and our algorithm recursively computes the set of samples at each time step. Since L_{t-1} reflects all previous observations, we can compute l_t using only L_{t-1} and o_t.

Figure 1 shows an overview of the algorithm. Initially, we assume the node has no knowledge about its position, so the initial samples are selected randomly from all possible locations. At each time step, the location set is updated based on possible movements and new observations. We estimate the location of the node by computing the average location of all possible locations in L_t. For our experiments, we assume locations are (x, y) positions in two dimensional Cartesian space, but the technique could be used equivalently for three dimensions or other location representations.

The steps are described in more detail in the following subsections. In the prediction step (Section 3.2), the node uses the transition distribution $p(l_t \mid l_{t-1})$ to predict its possible locations based on previous samples and its movement. In the filtering step (Section 3.3), the node uses new information received to eliminate predicted locations that are inconsistent with observations. Re-sampling is used to maintain the number of location samples.

3.2 Prediction

In the prediction step, a node starts from the set of possible locations computed in the previous step, L_{t-1}, and applies the mobility model to each sample to get a set of new samples, L_t. We assume a node is unaware of its moving speed and direction, other than knowing its speed is less than v_{max}. So, if in previous step l_{t-1}^i is one possible position of a node, the possible current positions are contained in the circular region with origin l_{t-1}^i and radius v_{max}. We use $d(l_1, l_2)$ to denote the Euclidean distance between two points l_1 and l_2. If speeds are distributed uniformly in the interval $[0, v_{max})$, the probability of current location based on previous location estimate is given by a uniform distribution:

$$p(l_t \mid l_{t-1}) = \begin{cases} \dfrac{1}{\pi v_{max}^2} & \text{if } d(l_t, l_{t-1}) < v_{max} \\ 0 & \text{if } d(l_t, l_{t-1}) \geqslant v_{max} \end{cases}$$

Hence, the set R computed in the prediction phase contains one location selected randomly from the circle of radius v_{max} around every point in L_{t-1}. This reflects the increased uncertainty about the node's location because of unknown motion. In cases where something is known about the node's motion (for example, that it is moving at a particular speed, or that it is more likely to be moving in a certain direction), the probability distribution can be adjusted accordingly to make better predictions.

3.3 Filtering

In this step, the node filters the impossible locations based on new observations. For simplification of presentation and analysis, we assume that time is discrete and all messages are received instantly. Hence, at time t, every node within radio range of a seed will hear a location announcement from that seed. In a realistic deployment, it would be necessary to deal with network collisions and account for missed messages.

Figure 2 shows an example situation. There are four types of seeds to consider:

outsiders – seeds that were not heard in either the current or the previous time quanta.

arrivers – seeds that were heard in the current time quantum, but not in the previous one.

leavers – seeds were heard in the previous time quantum, but not in this one.

insiders – seeds that were heard in both time quanta.

Arrivers and leavers provide the most useful information since the node will know it was within distance r of l_0 at time t_0, but not within distance r of l_1 at time t_1. If we only rely on direct information from seeds, however, a node will not know the previous location of an arriver, or the current location of a leaver. There are two possible ways to gather this information:
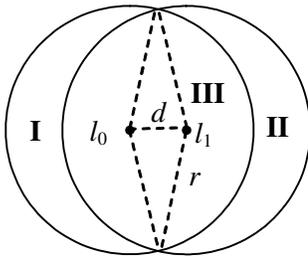
1. A seed node (S) transmits both its current location and its location at the previous time step in each announcement:

    $S \rightarrow$ Region   HELLO | ID_S | loc_t | loc_{t-1}

2. Neighbor nodes can transmit information about seed locations (the set of all seeds and their locations heard in the previous time step):

    $S \rightarrow$ Region   HELLO | ID_S | loc_t
    $N \rightarrow$ Region   HELLO | ID_N | {(ID_S, loc_{S_t}) }

The second approach is more expensive, but its cost may be combined with neighbor discovery in applications that require neighborhood information for other purposes. The advantage of the second approach is it also allows nodes to discover information about outsider seeds without keeping track of arrivers and leavers. The node knows it is not within distance r of any outsider seed, but must be within distance $2r$ of any seed heard by one of its neighbors.

Figure 3 shows the filter condition for insider and outsider seeds. Let S denote the set of all seeds heard by N and T denote the set of all nodes heard by N's neighbors but not by N. Then the filter condition of location l is

$$filter(l) = \forall s \in S, \, d(l, s) \leq r \wedge \forall s \in T, \, r < d(l, s) \leq 2r.$$
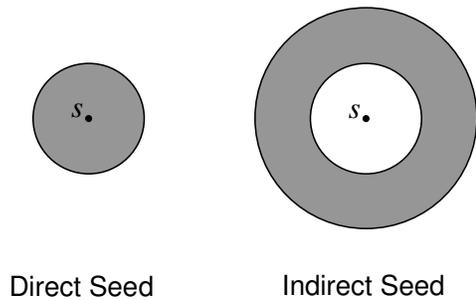
The probability distribution $p(l_t \mid o_t)$ is zero if the filter condition is false, and evenly distributed otherwise. Thus, we eliminate locations that are inconsistent with observations from the possible location set. After filtering, there may be fewer than N possible locations remaining. The prediction and filtering processes repeat, unioning the possible points found, until at least N possible locations have been acquired. (Section 5.6 considers how the choice of N affects accuracy.)

4. ANALYSIS

In this section we justify the importance sampling and re-sampling approaches used in our algorithm, and analyze the accuracy of location estimates produced. We conclude by discussing security issues when localization is used in hostile environments.

4.1 Importance sampling

We are interested in estimating the posterior distribution of the node's location $p(l_t \mid o_0, o_1, \ldots, o_t)$. Since it is generally impossible to sample from the posterior distribution directly, our algorithm adopts an importance sampling approach. Suppose samples are drawn independently from a normalized importance function . Then, we can measure the weight of each sample and use these weights to estimate

Figure 2. Seed Movement. The seed moves from l_0 at time 0 to position l_1 on time 1. The seed is an *insider* for nodes in region III, an arriver for nodes in region II, a leaver for nodes in region I, and an outsider for all other nodes.

Direct Seed   Indirect Seed

Figure 3. Filter condition. A node is within distance r if a seed it hears directly, and within distance (r, $2r$] of a seed it doesn't hear itself but one of its neighbors hears.

the posterior distribution.

As introduced in Figure 1, we adopt the following recursive importance function from [13]:

$$(l_t \mid o_0, o_1, \ldots, o_t) = p(l_0) \prod_{k=1}^{t} p(l_k \mid l_{k-1}) \quad (1)$$

$$\widetilde{w}_t^i = \widetilde{w}_{t-1}^i \, p(o_t \mid l_t^i) \quad (2)$$

$$w_t^i = \frac{\widetilde{w}_t^i}{\sum_{k=1}^{N} \widetilde{w}_t^k} \quad (3)$$

Equation (1) is the prediction phase, in which a node predicts its current possible locations based on previous possible locations. Equation (2) is the update phase, in which a node updates the weights of the new samples based on the received observations. Then, we normalize the weights $\widetilde{w}_t^i$ to w_t^i in (3) and use the weighted set $\left(l_t^i, w_t^i\right)$ to simulate the posterior distribution.

We choose this importance function because it yields the recursive calculation of $p(l_k \mid l_{k-1})$ and $p(l_t \mid o_t)$ in equations (1) and (2), and the distributions of the two probabilities are not hard to calculate, as shown in Section 3.2 and 3.3, respectively.

As shown in [16], under weak assumptions the importance sampling converges to the posterior distribution.

4.2 Re-sampling
The degeneracy of the importance sampling is unavoidable since the unconditional variance of the importance weights will increase [27]. The basic idea of re-sampling is to eliminate trajectories that have small normalized importance weights and to concentrate upon trajectories with large weights. A suitable measure of degeneracy of the algorithm is the effective sample size N_{eff} [27]. An estimate of N_{eff} is given by:

$$N_{eff} \approx \frac{1}{\sum_{i=1}^{N} (w_t^i)^2} \quad (4)$$

When N_{eff} is below a fixed threshold $N_{threshold}$, the re-sampling method needs to be used.

In our algorithm, since $p(o_t \mid l_t)$ is either 1 or 0, the weight is always 0 or 1. Assuming there are k 1s in the un-normalized N samples, then the normalized weights will consist of k values of $1/k$ and $n\text{-}k$ values of 0. From equation (4), we can compute

$$N_{eff} = \frac{1}{\sum_{i=1}^{N} (w_t^i)^2} = k$$

So in our algorithm, N_{eff} is exactly the number of valid samples. We only need to keep enough valid samples (samples with weight 1) to make N_{eff} equal to or above $N_{threshold}$. In our algorithm, we set $N_{threshold} = N$, and we repeatedly draw samples until we get exactly N valid ones.. In our simulation experiments described in Section 5.6, we find that N=50 is sufficient.
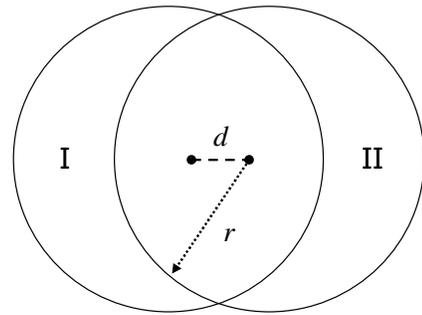
4.3 Resolution limit
Suppose nodes are randomly distributed in the network. As analyzed in [34], there is a theoretical limit on resolution in any range-free algorithm that is based only on connectivity.

The probability that a node can move a distance d without changing connectivity is exactly the probability there are seeds that are arrivers or leavers because of this move. In Figure 4, it is exactly the probability that there are no nodes in regions I and II. The area of region I and II is approximately $4rd$.

If the area of the region that can affect connectivity is krd, then and the maximum distance a sensor can move without changing connectivity is $\pi r/kn$ where n is the network density [34].

In our algorithm we consider both one hop and two hop seeds so a position difference would be noticeable if it affects either the one-hop or two-hop connectivity. If we assume the node density is high but the seed density is low, then a position change of distance d is noticeable if there are any seeds within the two-hop area reachable from this position, but not from the other position. The area of that region is approximately $4rs_d+4(2r)s_d = 12rs_d$. This assumes there is a high density of nodes, so the node has a high likelihood of having a neighbor that can hear the new seed. So, the resolution limit (maximum distance a sensor can move without changing connectivity) is $\pi r/12s_d$, where s_d is the seed density. For $s_d =1$, the resolution limit is $0.26r$.

Figure 4. Resolution Limit. Regions I and II are regions that can affect node's connectivity when it moves d distance.

4.4 Security

Although extending our technique to handle hostile environments is beyond the scope of this paper, MCL has several properties which make it well suited for adaptation into a secure localization protocol. Secure localization is a challenging problem that has not yet been studied.

Previous localization techniques are vulnerable to several kinds of attacks, and an attacker may be able to disrupt the integrity or availability of all known localization techniques. Without authentication, an attacker may inject bogus seeds into the network and announce false locations. Techniques like Centroid which estimate the location by averaging the locations of all seeds heard are susceptible to this attack.

Digital signatures can prevent bogus seeds from injecting bogus location messages by authenticating seeds' transmissions to nodes. This could be done by distributing public keys corresponding to the seeds' private keys to each node before deployment. Public key encryption operations are often too computationally expensive for sensor nodes, however, and the long messages required drain power resources. Another approach would be to use the μTesla protocol [38], by preloading each node with the initial hash chain value and each seed with the initial secret. This would save the expense of public key operations, but would delay localization (or at least verification of locations) until the next key in the hash chain is released. It would also require loose synchronization among the seeds.

Alternative key establishment mechanisms such as random key predistribution [14, 9] can establish keys using symmetric cryptography. These techniques require bidirectional messages, however, and are not suitable for localization techniques where seeds transmit further than nodes. Further, they would require the seed to transmit its location directly to each neighbor node, instead of just using a single transmission that can be heard and decoded by all nodes.

Note that hop count based algorithms [35, 34] require that the hop count is maintained accurately throughout the network and empower a single rogue node to disrupt many node locations by advertising a false hop count. If authentication is used to mitigate this threat, it requires link keys to be established between all communicating nodes. If a single node is compromised, locations throughout the network will be disrupted.

Localization protocols are particularly susceptible to replay attacks. In a wormhole attack, an attacker obtains two transceivers in the network connected by a high quality out-of-band link and replays messages heard at one location at the other location [22, 24]. A wormhole attack can easily disrupt existing localization techniques (including MCL), even if all location announcements were successfully encrypted. Known defenses for wormhole attacks rely on either nodes already knowing their locations [22], or require specialized hardware such as tightly synchronized clocks [22] or directional antennas [21].

MCL offers several desirable properties for security adaptation. Since it does not depend on seeds with powerful transmitters, bidirectional verification and key establishment is possible. When nodes and seeds move, an attacker will only be able to do limited damage. The MCL filtering approach is less susceptible to rogue seeds than other approaches. As long as there is valid information received from some legitimate seeds, the estimated locations will be bounded by the filtering of all valid information. The adversary can at worst prevent the node from acquiring any valid samples, and hence determining anything about its location. This is an effective service denial attack, but does not compromise the integrity of location results. In addition, since MCL continually updates location estimates, an attack will only be successful as long as it continues to transmit bogus messages.

5. EVALUATION

The key metric for evaluating a localization technique is the accuracy of the location estimates versus the communication and deployment costs. Increasing the density of seeds or the frequency of location announcements should improve accuracy, but the tradeoffs need to be understood to determine appropriate deployment parameters. In this section, we evaluate the MCL technique by measuring how its estimated location errors vary with various network and algorithm parameters described in Section 5.1.

In addition, we compare our results to those for other range-free localization techniques, namely the Amorphous [34] and Centroid [6] techniques described in Section 2.

Our simulation experiments were conducted using a purpose-built simulator which is available from http://www.cs.virginia.edu/mcl. Experimental results are the average of 10 executions with different pseudorandom number generator seeds.

5.1 Simulation Parameters

In our experiments, we vary parameters of both the sensor network and sensor nodes, and of the MCL algorithm.

For all of our experiments, sensor nodes are randomly distributed in a 500m x 500m rectangular region. We assume a fixed transmission range, r, of 50m for both nodes and seeds. The network and node parameters we vary are:

- Speed of the nodes and seeds (v_{max}, v_{min}, s_{max}, s_{min}). We represent the speed as the moving distance per

time unit. A node's speed is randomly chosen from $[v_{min}, v_{max}]$; a seed's speed is randomly chosen from $[s_{min}, s_{max}]$. We consider the impact of speeds on both accuracy and convergence time.

- Node density (n_d), the average number of nodes in one hop transmission range. We study the effects of varying n_d in Section 5.5, and use a fixed $n_d = 10$ for other experiments.
- Seed density (s_d), the average number of seeds in one hop transmission range (considered in Section 5.4).

We adopt the random waypoint mobility model [8] for both nodes and seeds. It is one of the most commonly used mobility models for mobile ad hoc networks. In the random waypoint model, a node randomly chooses its destination, its speed of movement, and its pause time after arriving at the destination. We assume nodes are unaware of their velocity and direction, but have a known maximum velocity v_{max}. As pointed out in [44], the random waypoint model suffers from the decay of average speed, and this will provide an unsound basis for simulation. We used a modified random waypoint model to maintain the average speed. Instead of choosing a certain speed for each destination, nodes randomly vary their speed during each movement. The pause time is set to 0, so the average speed is exactly $v_{max}/2$ when speed is chosen randomly between 0 and v_{max}. In Section 5.8, we consider how different mobility models affect the localization accuracy.

We assume a node can judge if it is within radio range r of another node or not, but it can not get more precise distance information (for example, measuring distance through received radio signal strength). For most of the experiments we model radio range as a perfect circle. This model is not realistic, however, and in Section 5.7 we consider the impact of irregularity on location estimates.

The MCL algorithm parameters we vary are:

- Time between location announcements (t_u). In most of our experiments we assume a fixed t_u and measure speeds in terms of r distance units per t_u. In Section 5.3, we consider the effect of varying node speed.
- Number of samples maintained (N). Keeping more samples improves accuracy but requires more memory and computation. In Section 5.6, we consider how location accuracy varies with the number of samples. Our results show that a few samples are sufficient for high accuracy. For all the other experiments, we use $N = 50$.
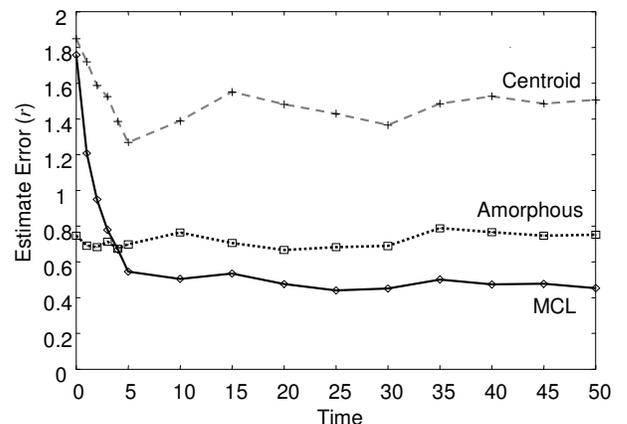
5.2 Accuracy

The accuracy of MCL depends on the speeds of the seeds and nodes. As time passes, nodes will receive more seed location announcements and improve their location estimates.
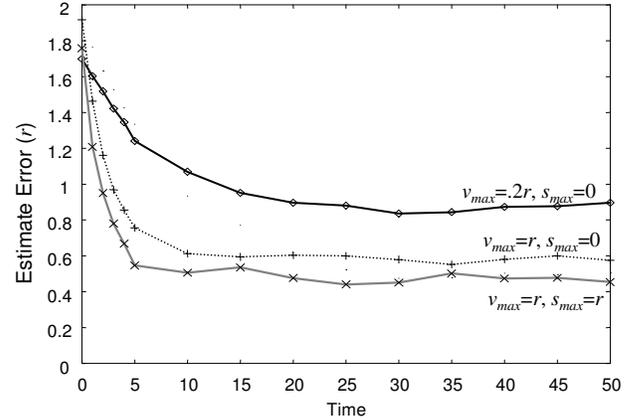
Figure 5. Location convergence. $n_d = 10$, $s_d = 1$.

Figure 5 shows the error in the location estimate, measured as a multiple of the node transmission distance r, for three different scenarios: stationary seeds ($s_{max} = 0$) with nodes moving with $v_{max} = .2r$ and r, and both nodes and seeds moving with $v_{max} = s_{max} = r$. The localization process can be divided into the initialization phase and the stable phase. In the initialization phase, the estimate error decreases dramatically as new observations are incorporated. The localization is improved by both the current observation and previous observations. In the stable phase, the impact of observations (filter) and the node's mobility (uncertainty) reach some balance, and the estimate error fluctuates around a minimum value. The faster the speed of the seeds and nodes, the quicker the stable phase is reached. The post-convergence accuracy is also better for faster moving nodes, since by moving quickly they encounter more seeds and more rapidly filter our inaccurate samples.

Unlike the MCL technique, the Centroid and Amorphous localization techniques do not exploit past information, so they do not improve over time. Figure 6 compares the localization error of different localization techniques over

Figure 6. Accuracy Comparison. $n_d = 10$, $s_d = 1$, $v_{max} = s_{max} = r$.

time. The accuracy of MCL improves quickly. After 5 steps, it exceeds the accuracy of the Amorphous technique.

5.3 Node Speed

Varying node speed is similar to varying the time between location announcements. If announcements are more frequent, localization is more accurate but communication overhead increases. We measure maximum node speed as v_{max} and distribute actual node speeds between 0 and v_{max} using the modified random waypoint mobility model. Figure 7 shows the impact of node speed on the converged localization error as the distance traveled per announcement time unit increases from $0.1r$ to $2r$ for a few different seed densities and seed velocities. Node speed impacts the localization process in two ways. The increased speed makes the predicted locations less accurate since the next possible locations fall into a larger region. On the other hand, faster movement leads to more new observations in each time step, and hence more impossible locations can be filtered. The estimate errors drop fast as node speeds increase from $0.1r$ to $0.3r$ when the seeds are also moving at the same speed, and then the error gradually increases as the uncertainty resulting from faster movement increases. With fixed velocity seeds ($s_{min} = s_{max} = r$), the error is least when nodes are slowest, and increases gradually as node speed increases.

Figure 7 illustrates that the length of time unit can be increased without increasing estimate error as long as the node's new location is within r distance of its previous location. MCL performs best when the maximum distance nodes and seeds travel from previous location per time unit is between $0.4r$ to r distance. Hence, in a network with slow moving nodes, the time unit can be quite long and communication costs are low. Communication costs are analyzed more thoroughly in Section 5.9.
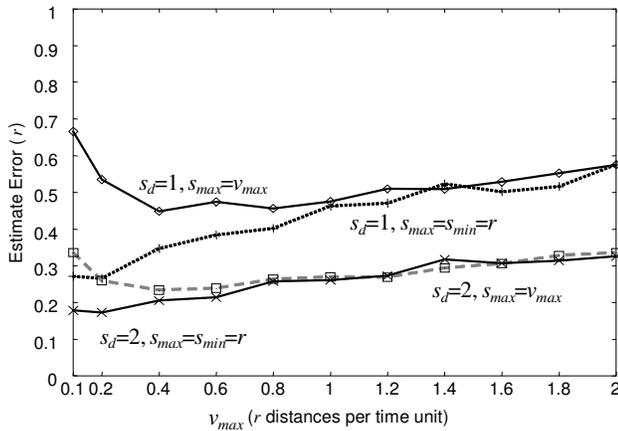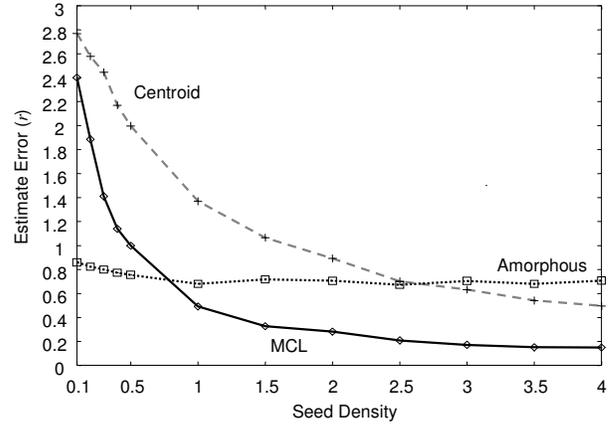

Figure 8. Impact of seed density. $n_d = 10$, $v_{max} = s_{max} = .2r$.

5.4 Seed Density

Increasing the density of seeds makes localization easier, but increases network and deployment costs. Figure 8 shows the average estimate error of different localization algorithms when seed density varies. The accuracy of both MCL and Centroid improves as seed density increases since nodes will receive more location announcements. For the Amorphous technique, since each node receives the propagated messages from all seeds in the network, the estimate error does not improve much after there are a sufficient number of seeds (32 in this experiment). MCL performs adequately even for low seed densities and outperforms the other techniques when seed density is 1 or above. Since the possible location set accounts for previous information about the node's location, MCL is much more accurate than Centroid when seed density is low.

5.5 Node Density

Figure 9 shows the impact of node density on estimate error in different localization algorithms. MCL and Centroid are little affected by node density. MCL requires a threshold node density in order for nodes to receive two-hop information from enough neighbors, but a few neighbors is
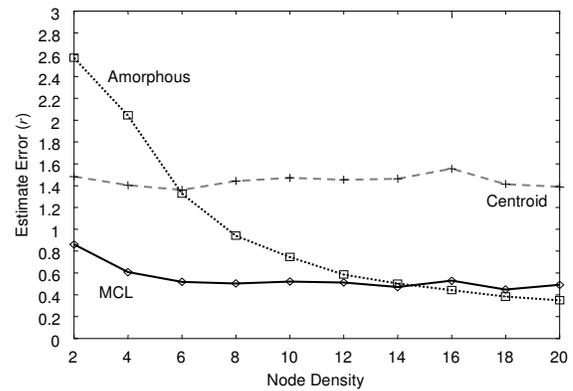

Figure 7. Impact of node speed.

Figure 9. Impact of node density. $s_d = 1$, $v_{max} = s_{max} = .2r$.

sufficient. The Amorphous technique depends on higher network density. It performs poorly when network density is below 6, but performs best when network density is larger than 15. This is because network density has great impact on the accuracy of hop count. [34] and [19] suggest approaches for improving hop counting based techniques when the node density is low by increasing the number of seed nodes.

5.6 Number of Samples

Maintaining more samples for the MCL algorithm can improve accuracy, but requires additional memory. Figure 10 shows the impact of sample size on location accuracy. The estimate error drops rapidly at the beginning, since a small number of samples cannot adequately reflect the probability distribution. The estimate error is fairly stable after sample size 50 and the accuracy improves only minimally by increasing the number of samples to 1000.

Hence, MCL is efficient in both memory and computation. Good accuracy is achieved with only 50 samples. The seed density also has an impact on the selection of sample size. The higher seed density, the more accurate the localization can be, so more samples are required to achieve minimal error. For applications with low seed density and severe computation and memory limits, the sample size can be reduced to 10 with little accuracy loss.

5.7 Irregularity

Variability in actual radio transmission patterns can have a substantial impact on localization accuracy depending on the localization technique. Unlike the prefect circles of radius r assumed in our previous experiments, the measured reception distance of radios can vary substantially with environmental conditions and antenna irregularities.

Figure 11 shows the impact of degree of irregularity on estimate error. The MCL and Centroid techniques are not
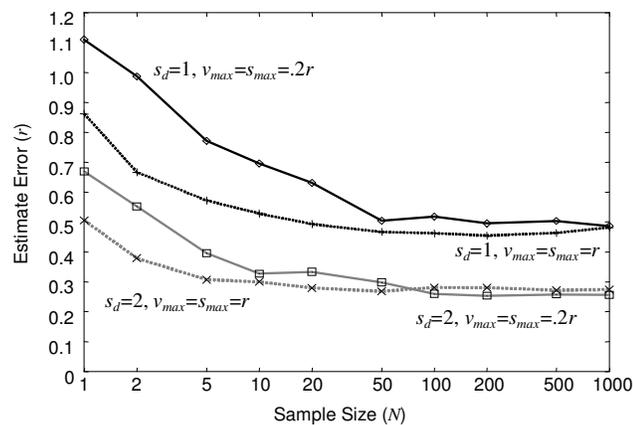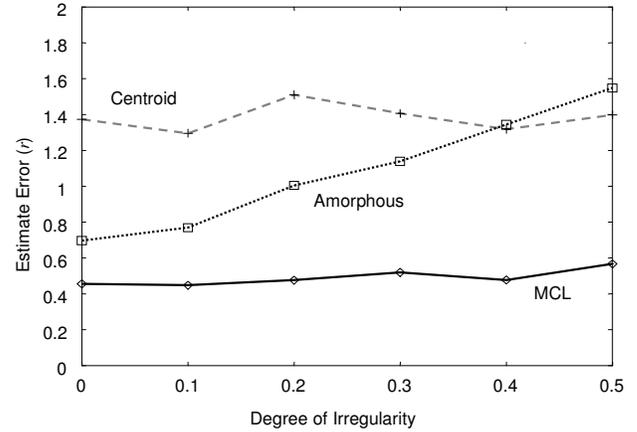
Figure 10. Impact of Sample Size. $n_d = 10$

Figure 11. Impact of Irregularity.
$n_d = 10$, $s_d = 1$, $v_{max} = s_{max} = .2r$

substantially affected. We use *degree of irregularity* (DOI) to denote the maximum radio range variation in the direction of radio propagation. For example, if DOI = 0.1, then the actual radio range in each direction is randomly chosen from [0.9r, 1.1r]. The estimate error of the Amorphous technique increases significantly as DOI increases. This is because the ranging estimate is from the propagation of hop count, and it always selects the minimal hop count. Hence, irregular radio transmissions will have an accumulating affect on the ranging error in multiple hops.

5.8 Motion Model

So far, we have assumed that both nodes and seeds move randomly and independently. In some applications, the motion of nodes and seeds may be correlated and demonstrate some group behavior, and this may affect the performance of our algorithm.

We use the Reference Point Group Mobility model (RPGM) [15] to investigate the effect of group behavior on our algorithm. In RPGM, the motion of a node is the combination of a group motion vector and a random motion vector. The random motion is based on a reference point that moves according to the group motion. This provides an approximation for a group of nodes and seeds moving in a current or being blown by the wind.

We put all nodes and seeds in the same group. The group motion is defined as a random walk model [Camp02], in which the direction is chosen randomly between 0 and 360 degrees and the speed is chosen randomly between 0 and the maximum group motion speed. Each node's individual random movement relative to the group motion is selected using the modified random waypoint model as in previous experiments. To maintain the same group motion for all
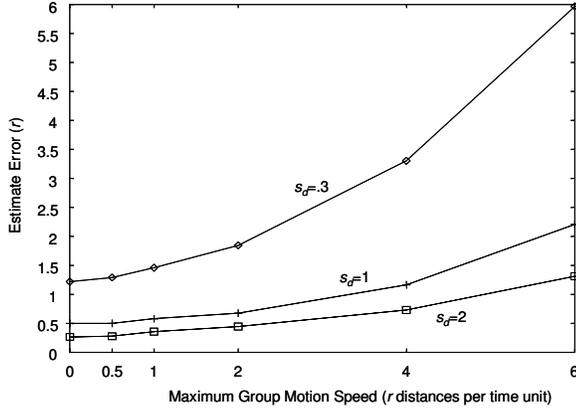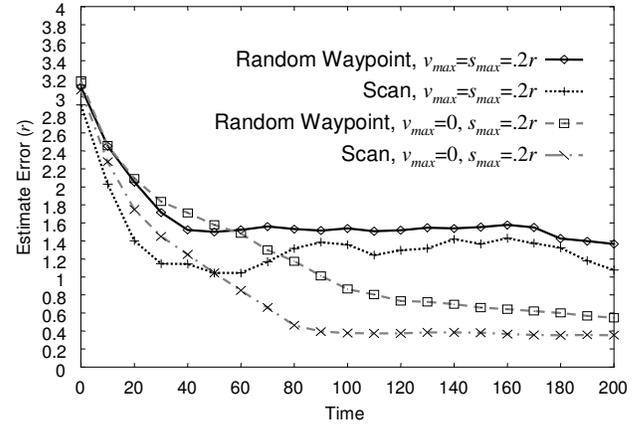
Figure 12. Impact of group motion.
$n_d = 10$, $v_{max} = s_{max} = r$, unbounded area.

Figure 13. Impact of motion control. $n_d = 10$, $s_d = 0.3$.

nodes, we assume there are no boundaries in the network so nodes can move freely. If a node cannot find enough valid samples after filtering, it will reinitialize itself by eliminating previous samples and drawing samples from new observations directly. We also assume a node is aware of the maximum distance it can move in one time unit, which is the sum of the maximum individual random movement and the maximum group motion.

Figure 12 shows the location accuracy when we keep the maximum random motion speed at r per time unit and vary the maximum group motion speed. The estimate error increases as the maximum group motion speed increases. Since all nodes are moving in the same way, the relative positions change less, so the number of useful new observations received does not increase with increasing group speed. Because the uncertainty in the prediction phase becomes larger as group motion speed increases, accuracy is substantially reduced when the group motion dominates the individual node movement.

On the other hand, in some applications it may be possible to control how seeds move. A strategy that moves seeds in a way to cover the area thoroughly will improve the accuracy, and especially the convergence time, of MCL.

Figure 13 shows how control over seed motion can improve the accuracy and convergence time of MCL localization. We consider both the static nodes and moving nodes scenarios here. When nodes are static, the prediction phase does not increase uncertainty so the estimate error decreases as more observations are collected. When nodes are mobile, the estimate error converges to balance the motion uncertainty and observations. We use a low seed density ($s_d = 0.3$) to make the localization process slow. We compare the random waypoint model with a scan model. In the scan model, seed nodes are evenly distributed and separated by $2r$. They scan the network in a predefined path that maximizes coverage. Both the convergence time and estimate error are reduced.

5.9 Communication Overhead
We measure the communication overhead as the number of messages a node needs to send in each localization process. Although the size of messages may vary slightly across localization techniques, the actual message size is more a function of how locations and announcements are encoded than the localization technique.

Since the Amorphous algorithm requires all seed information to be flooded to the network, each node needs to broadcast exactly the number of seeds in each localization process. For reasonable location accuracy, there are 32 seeds and this means every node needs to retransmit 32 location announcements. For the Centroid algorithm, the nodes do not need to transmit any messages; all location transmission come directly from the seeds. For the MCL algorithm, nodes must share their seed information with their one-hop neighbors, so in each time step all nodes transmit their seed information once. The number of seeds each node hears is a function of the seed density. In our experiments, a seed density of one ($s_d = 1$) is adequate for precise localization, so the communication overhead of each node is exactly 1.

6. CONCLUSION
Many wireless sensor network applications depend on nodes being able to accurately determine their locations. This is the first work to study range-free localization in the presence of mobility. Our main result is surprising and counterintuitive: mobility can improve the accuracy and reduce the costs of localization. Our simulation experiments reveal that the MCL technique can provide accurate localization even when memory limits are severe, the seed density is low, and network transmissions are

highly irregular. Many issues remain to be explored in future work including how well our assumptions hold in different mobile sensor network applications, how different types of motion affect localization, and how our technique can be extended to provide security.

ACKNOWLEDGEMENTS

REFERENCES

[1] Jeffrey Antol, Philip Calhoun, John Flick, Gregory A. Hajos, Robert Kolacinski, David Minton, Rachel Owens and Jennifer Parker. *Low Cost Mars Surface Exploration: The Mars Tumbleweed.* NASA Langley Research Center. NASA/TM-2003-212411. August 2003.

[2] Paramvir Bahl and Venkata N. Padmanabhan. RADAR: An In-Building RF-Based User Location and Tracking System. *IEEE InfoCom 2000*, March 2000.

[3] P. Bergamo and G. Mazzini. Localization in Sensor Networks with Fading and Mobility. *IEEE PIMRC*. September 2002.

[4] Wolfram Burgard, Dieter Fox, Daniel Hennig and Timo Schmidt. Estimating the Absolute Position of a Mobile Robot Using Position Probability Grids. *14th National Conference on Artificial Intelligence* (AAAI). 1996.

[5] Wolfram Burgard, Andreas Derr, Dieter Fox, and Armin B. Cremers. Integrating Global Position Estimation and Position Tracking for Mobile Robots: The Dynamic Markov Localization Approach. *IEEE/RSI International Conference on Intelligence Robots and Systems* (IROS). 1998.

[6] Nirupama Bulusu, John Heidemann and Deborah Estrin. GPS-less Low Cost Outdoor Localization for Very Small Devices. *IEEE Personal Communications Magazine*. October 2000.

[7] Nirupama Bulusu, John Heidemann and Deborah Estrin. Density Adaptive Algorithms for Beacon Placement in Wireless Sensor Networks. *IEEE ICDCS 2001*. April 2001.

[8] Tracy Camp, Jeff Boleng and Vanessa Davies. A Survey of Mobility Models for Ad Hoc Networks Research. *Wireless Communications and Mobile Computing*. Volume 2, Number 5. 2002.

[9] Haowen Chan, Adrian Perrig and Dawn Song. Random Key Predistribution Schemes for Sensor Networks. *IEEE Symposium on Security and Privacy*. May 2003.

[10] Frank Dellaert, Dieter Fox, Wolfram Burgard and Sebastian Thrun. Monte Carlo Localization for Mobile Robots. *IEEE International Conference on Robotics and Automation* (ICRA). May 1999.

[11] Lance Doherty, Kristofer Pister and Laurent El Ghaoui. Convex Position Estimation in Wireless Sensor Networks. *IEEE InfoCom 2001*. April 2001.

[12] Arnaud Doucet, Simon. Godsill and Christophe Andrieu. On Sequential Monte Carlo Sampling Methods for Bayesian Filtering. *Statistics and Computing*. Volume 10, pp. 197-208. 2000.

[13] Arnaud Doucet, Nando de Freitas and Neil Gordon. An Introduction to Sequential Monte Carlo Methods. In *Sequential Monte Carlo Methods in Practice*, eds. Arnaud Doucet, Nando de Freitas and Neil Gordon. 2001.

[14] Laurent Eschenauer and Virgil D. Gligor. A Key-Management Scheme for Distributed Sensor Networks. *9th ACM Conference on Computer and Communication Security*. November 2002.

[15] Xiaoyan Hong, Mario Gerla, Guangyu Pei and Ching-Chuan Chiang. A Group Mobility Model for Ad Hoc Wireless Networks. *ACM International Workshop on Modeling and Simulation of Wireless and Mobile Systems (MSWiM)*. August 1999.

[16] John Geweke. Bayesian Inference in Econometric Models Using Monte Carlo Integration. *Econometrica*. Volume 57, Number 6. 1989.

[17] Neil J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel Approach to Nonlinear/Non-Gaussian Bayesian State Estimate. *IEE Proceedings*. Volume 140, pp. 107-113. 1993.

[18] J. E. Handschin. Monte Carlo Techniques for Prediction and Filtering of Non-Linear Stochastic Processes. *Automatica* 6. pp. 555-563.1970.

[19] Tian He, Chengdu Huang, Brian M. Blum, John A. Stankovic, Tarek Abdelzaher. Range-free Localization Schemes for Large Scale Sensor Networks. *MobiCom* 2003.

[20] Jeffrey Hightower and Gaetano Borriello. Location Systems for Ubiquitous Computing. *IEEE Computer*. Vol 34, No. 8. August 2001.

[21] Lingxuan Hu and David Evans. Using Directional Antennas to Prevent Wormhole Attacks. *Network and Distributed System Security Symposium* (NDSS), February 2004.

[22] Yih-Chun Hu, Adrian Perrig and David Johnson. Packet Leashes: A Defense against Wormhole Attacks

in Wireless Ad Hoc Networks. *IEEE InfoCom* 2003. April 2003.

[23] Michael Isard and Andrew Blake. Contour Tracking by Stochastic Propagation of Conditional Density. *European Conference on Computer Vision*, pp. 343-356. 1996.

[24] Chris Karlof and David Wagner. Secure Routing in Sensor Networks: Attacks and Countermeasures. *First IEEE International Workshop on Sensor Network Protocols and Applications*, May, 2003.

[25] Brad Karp and H. T. Kung. Greedy Perimeter Stateless Routing. *MobiCom* 2000.

[26] Young-Bae Ko and Nitin H. Vaidya. Location-Aided Routing (LAR) in Mobile Ad Hoc Networks. *MobiCom* 1998.

[27] A. Kong, J. S. Liu and W. H. Wong. Sequential Imputations and Bayesian Missing Data Problems. *Journal of the American Statistical Association.* Volume 89, pp. 278-288. 1994.

[28] Uwe Kubach and Kurt Rothermel. Exploiting Location Information for Infostation-Based Hoarding. *MobiCom* 2001.

[29] Andrew M. Ladd, Kostas E. Bekris, Guillaume Marceau, Algis Rudys, Lydia E. Kavraki and Dan S. Wallach, Robotics-Based Location Sensing using Wireless Ethernet. *MobiCom* 2002.

[30] Andrew M. Ladd, Kostas E. Bekris, Algis P. Rudys, Dan S. Wallach and Lydia E. Kavraki. On the Feasibility of Using Wireless Ethernet for Indoor Localization. *IEEE Transactions on Robotics and Automation*. Volume 20, Number 3. June 2004.

[31] Kenneth Lodding. Personal communication. March 2004.

[32] Martin Mauve, Jörg Widmer and Hannes Hartenstein. A Survey on Position-Based Routing in Mobile Ad-Hoc Networks. *IEEE Network Magazine*. 2001.

[33] Peter Maybeck. *Stochastic Models. Estimation and Control, Volume 1.* Academic Press, New York, 1979.

[34] Radhika Nagpal, Howard Shrobe, and Jonathan Bachrach. Organizing a Global Coordinate System from Local Information on an Ad Hoc Sensor Network. *2nd International Workshop on Information Processing in Sensor Networks* (IPSN). April 2003.

[35] Dragos Niculescu and Badri Nath. DV Based Positioning in Ad hoc Networks. *Kluwer Journal of Telecommunication Systems*. 2003.

[36] Dragos Niculescu and Badri Nath. Ad Hoc Positioning System (APS) Using AoA. *IEEE InfoCom* 2003.

[37] Neal Patwari and Alfred O. Hero III. Using Proximity and Quantized RSS for Sensor Localization in Wireless Networks. *Workshop on Wireless Sensor Networks and Applications*. September 2003.

[38] Adrian Perrig, Robert Szewczyk, Victor Wen, David Culler, and Doug Tygar. SPINS: Security Protocols for Sensor Networks. *Wireless Networks Journal* (WINE). September 2002.

[39] D. B. Rubin. Using the SIR algorithm to simulate posterior distributions. *Bayesian Statistics* 3. Oxford University Press. 1988.

[40] Andreas Savvides, Chih-Chieh Han, Mani B. Strivastava. Dynamic fine-grained localization in Ad-Hoc networks of sensors. *MobiCom* 2001.

[41] Yi Shang, Wheeler Ruml, Ying Zhang, Markus Fromherz. Localization from Mere Connectivity. *MobiHoc 2003*. June 2003.

[42] Sebastian Thrun, Dieter Fox, Wolfram Burgard and Frank Dellaert. Robust Monte Carlo Localization for Mobile Robots. *Artificial Intelligence Journal*. 2001.

[43] B. H. Wellenhoff, H. Lichtenegger and J. Collins. *Global Positioning System: Theory and Practice, Fourth Edition.* Springer Verlag. 1997.

[44] Jungkeun Yoon, Mingyan Liu and Brian Noble. Sound Mobility Models. *MobiCom* 2003.