MoteTrack: A Robust, Decentralized Approach to RF-Based Location Tracking

Konrad Lorincz and Matt Welsh

Harvard University
Division of Engineering and Applied Sciences
Cambridge MA 02138, USA
{konrad, mdw}@eecs.harvard.edu
http://www.eecs.harvard.edu/~konrad/projects/motetrack

Abstract. In this paper, we present a robust, decentralized approach to RF-based location tracking. Our system, called MoteTrack, is based on low-power radio transceivers coupled with a modest amount of computation and storage capabilities. MoteTrack does not rely upon any back-end server or network infrastructure: the location of each mobile node is computed using a received radio signal strength signature from numerous beacon nodes to a database of signatures that is replicated across the beacon nodes themselves. This design allows the system to function despite significant failures of the radio beacon infrastructure. In our deployment of MoteTrack, consisting of 20 beacon nodes distributed across our Computer Science building, we achieve a 50th percentile and 80th percentile location-tracking accuracy of 2 meters and 3 meters respectively. In addition, MoteTrack can tolerate the failure of up to 60% of the beacon nodes without severely degrading accuracy, making the system suitable for deployment in highly volatile conditions. We present a detailed analysis of MoteTrack's performance under a wide range of conditions, including variance in the number of obstructions, beacon node failure, radio signature perturbations, receiver sensitivity, and beacon node density.

1 Introduction

Using radio signal information from wireless transmitters, such as 802.11 base stations or sensor network nodes, it is possible to determine the location of a roaming node with close to meter-level accuracy [1, 2]. Such *RF-based location tracking systems* have a wide range of potential applications. We are particularly concerned with applications in which the *robustness* of the location-tracking infrastructure is at stake. For example, firefighters entering a large building often cannot see due to heavy smoke coverage and have no *a priori* notion of building layout. An RF-based location tracking system would allow firefighters and rescuers to use a heads-up display to track their location and monitor safe exit routes [3]. Likewise, an incident commander could track the location of multiple rescuers in the building from the command post. Such capabilities would have greatly improved FDNY rescue operations on September 11, 2001, according to the McKinsey reports [4].

We note that our system needs to be installed and calibrated before it can be used. We consider this part of bringing a building "up to code", similar to installing smoke detectors, fire and police radio repeaters in high-rise buildings, and other such safety devices. For scenarios where an offline calibration is infeasible (e.g. because the emergency is in a remote location such as a field, highway, etc.), our scheme as described in the paper is not appropriate. It remains an open research question how to address this issue, and we provide some suggestions in the future work section.

RF-based location tracking is a well-studied problem, and a number of systems have been proposed based on 802.11 [1, 5, 6, 2, 7] or other wireless technologies [8]. To date,

however, existing approaches to RF-based localization are *centralized* (i.e., they require either a central server or the user's roaming node, such as PDA or laptop, to compute the user's location) and/or use a *powered infrastructure*. In a fire, earthquake, or other disaster, electrical power, networking, and other services may be disabled, rendering such a tracking system useless. Even if the infrastructure can operate on emergency generator power, requiring wireless *connectivity* is impractical when a potentially large number of wireless access points may themselves have failed (e.g., due to physical damage from fire).

In addition, most previous approaches are *brittle* in that they do not account for lost information, such as the failure of one or more transmitters, or perturbations in RF signal propagation. As such, existing approaches are inappropriate for safety-critical applications, such as disaster response, in which the system must continue to operate (perhaps in a degraded state) after the failure of one or more nodes in the tracking infrastructure.

In this paper, we present a *robust, decentralized* approach to RF-based localization, called *MoteTrack*. MoteTrack uses a network of battery-operated wireless nodes to measure, store, and compute location information. Location tracking is based on empirical measurements of radio signals from multiple transmitters, using an algorithm similar to RADAR [1]. To achieve robustness, MoteTrack extends this approach in three significant ways:

- First, MoteTrack uses a decentralized approach to computing locations that runs on the programmable beacon nodes, rather than a back-end server.
- Second, the location signature database is replicated across the beacon nodes themselves in a fashion that minimizes per-node storage overhead and achieves high robustness to failure.
- Third, MoteTrack employs a dynamic radio signature distance metric that adapts to loss of information, partial failures of the beacon infrastructure, and perturbations in the RF signal.

In our deployment of MoteTrack, consisting of 20 beacon nodes distributed over one floor of our Computer Science building, we achieve a 50th percentile and 80th percentile location-tracking accuracy of 2 meters and 3 meters respectively, which is similar to or better than other RF-based location tracking systems. Our approach to decentralization allows MoteTrack to tolerate the failure of up to 60% of the beacon nodes without severely degrading accuracy, making the system suitable for deployment in highly volatile conditions. We present a detailed analysis of MoteTrack's performance under a wide range of conditions, including variance in the number of obstructions, beacon node failure, radio signature perturbations, receiver sensitivity, and beacon node density.

2   Background and Related Work

A number of indoor location tracking systems have been proposed in the literature, based on RF signals, ultrasound, infrared, or some combination of modalities. Our goal is to develop a system that operates in a decentralized, robust fashion, despite the failure of individual beacon nodes. This robustness is essential in order for the system to be used in disaster response, firefighting, or other critical applications in which a centralized approach is inappropriate.

As mentioned previously, RF-based location tracking has been widely studied [1, 9–11, 8, 12, 5, 6, 2]. Given a model of radio signal propagation in a building or other environment, received signal strength can be used to estimate the distance from a transmitter to a receiver, and thereby triangulate the position of a mobile node [13]. However,

this approach requires detailed models of RF propagation and does not account for variations in receiver sensitivity and orientation.

An alternative approach is to use empirical measurements of received radio signals to estimate location. By recording a database of radio "signatures" along with their known locations, a mobile node can estimate its position by acquiring a signature and comparing it to the known signatures in the database. A weighting scheme can be used to estimate location when multiple signatures are close to the acquired signature. All of these systems require that the signature database be collected manually prior to system installation, and rely on a central server (or the user's mobile node) to perform the location calculation.

Several systems have demonstrated the viability of this approach. RADAR [1] obtains a 75th percentile location error of just under 5 meters, while DALS [12] obtains an 87th percentile location error of about 9 meters. These basic schemes have also been extended to improve accuracy for tracking moving targets [9]. MoteTrack's basic location estimation uses a signature-based approach that is largely similar to RADAR. Our goal is not to improve upon the accuracy of the basic signature-based localization scheme, but rather to improve the robustness of the system through a decentralized approach.

Ultrasound-based systems, such as Cricket [14, 15] and the Active Bat [16], can achieve much higher accuracies using time-of-flight ranging. However, these systems require line-of-sight exposure of receiver to ultrasound beacons in the infrastructure, and may require careful orientation of the receiver. Such an approach is acceptable for infrequent use by unencumbered users in an office environment, but less practical for rescue workers. A multimodal system would be able to achieve high accuracy when ultrasound is available and well-positioned, and fall back on less-accurate RF signal strength otherwise. Infrared-based systems, including the Active Badge [17], can localize a user to a specific area with direct line-of-sight exposure to the IR beacon, but suffer errors in the presence of obstructions and differing light and ambient IR levels (as in a fire).

2.1   MoteTrack goals and challenges

We first define what we mean by *robustness* with respect to location tracking. Signature-based localization schemes require a set of base stations, generally at fixed locations, to either transmit periodic beacon messages or receive signals from mobile nodes. One form of robustness, then, is graceful degradation in location accuracy as base stations fail (say, due to fire, electrical outage, or other causes).

Another form of robustness is resiliency to information loss. For example, a mobile node may be unable to communicate with an otherwise active base station, due to poor antenna orientation, multipath fading, interference, or other (perhaps transient) effects. If the tracking system assumes complete information when comparing RF signatures, this partial information loss may lead to large errors.

A third type of robustness has to do with perturbations in RF signals between the time that the signature database was collected and the time that the mobile node is using this information to estimate location. Due to the movement of base stations, furniture, opening or closing of doors, and other environmental conditions, an RF signature may no longer be valid after it has been initially acquired. The tracking system should work well even in the presence of this kind of variance in the received RF signals.

The final type of robustness has to do with the location estimation computation itself. As mentioned before, most of the previous work in this area has employed a central server to collect RF signatures and compute a mobile node's location. This approach is clearly undesirable since this server is a single point of failure. Traditional fault-tolerance schemes, such as server failover, are still susceptible to large-scale outages of electrical power or the wired network infrastructure.
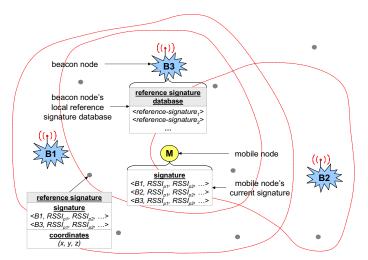
Fig. 1. The MoteTrack location system. $B1$, $B2$, *and* $B3$ *are beacon nodes, which broadcast beacon messages at various transmission powers (*$p1$, $p2$, *etc.). Each beacon node stores a subset of all reference signatures.* M *is a mobile node that can hear from all three beacon nodes. It aggregates beacon messages received over some time period into a signature. The areas marked by red perimeters indicate the reachability of beacon messages from the corresponding beacon node.*

Given these goals, a number of challenges arise that we wish to address through MoteTrack. First, the collection of RF signatures and location calculation must be resilient to loss of information and signal perturbation. This requires a signature distance metric that takes loss into account, avoiding explosion of error when one or more base stations cannot be contacted.

Another set of challenges has to do with decentralizing the location tracking system. One approach is to allow the base station nodes themselves to perform location estimation, rather than relying on a central server. This leads to questions about the required resources and cost of the base stations, and whether they can be readily programmed to provide this functionality. An alternative is to allow the mobile device to perform location estimation directly. In its simplest form, the entire RF signature database could be stored on the mobile node. In cases where a mobile user only carries a small RF beacon or listener (e.g., embedded into a firefighter's equipment), this may not be feasible.

3 MoteTrack Overview

In this section we give an overview of the MoteTrack system, shown in Figure 1. MoteTrack is based on low-power, embedded wireless devices, such as the Berkeley Mica2 sensor "mote." The advantages of this platform over traditional 802.11 base stations are that Mica2 motes are inexpensive, small, low-power, and (most importantly) *programmable* — we can easily push new programs and data to each device via their radio. However, the MoteTrack approach could be readily applied to other wireless networks based on 802.11, Bluetooth, or 802.15.4, given the ability to program base stations appropriately.

In MoteTrack, a building or other area is populated with a number of Mica2 motes acting as *beacon nodes*. Beacon nodes broadcast periodic *beacon messages*, which consist of a tuple of the format {*sourceID, powerLevel*}. *sourceID* is the unique identifier of the beacon node, and *powerLevel* is the transmission power level used to broadcast

the message. Each mobile node that wishes to use MoteTrack to determine its location listens for some period of time to acquire a *signature*, consisting of the set of beacon messages received over some time interval. Finally, we define a *reference signature* as a signature combined with a known three-dimensional location (x, y, z).

The location estimation problem consists of a two-phase process: an *offline* collection of reference signatures followed by *online* location estimation. As in other signature-based systems, the reference signature database is acquired manually by a user with a laptop and a radio receiver. Each reference signature, shown as gray dots in Figure 1, consists of a set of *signature tuples* of the form {*sourceID, powerLevel, meanRSSI*}. *sourceID* is the beacon node ID, *powerLevel* is the transmit power level of the beacon message, and *meanRSSI* is the mean received signal strength indication (RSSI) of a set of beacon messages received over some time interval. Each signature is mapped to a known location by the user acquiring the signature database.

In MoteTrack, beacon nodes broadcast beacon messages at a range of transmission power levels. Using multiple transmission power levels will cause a signal to propagate at various levels in its medium and therefore exhibit different characteristics at the receiver. In the most extreme case, a slight increase in the transmission power may make the difference between whether or not a signal is heard by a receiver. Varying transmission power therefore diversifies the set of measurements obtained by receiving nodes and in fact increases the accuracy of tracking by several meters in our experiments (see Section 6.4).

3.1 Location estimation

Given a mobile node's received signature s and the reference signature set R, the mobile node's location can be estimated as follows. (In this section, we discuss the approach as though it were centralized; in Section 4 we present our decentralized design.) The first step is to compute the *signature distances*, from s to each reference signature $r_i \in R$. We employ the Manhattan distance metric,

$$M(r, s) = \sum_{t \in T} |meanRSSI(t)_r - meanRSSI(t)_s|$$

where T is the set of signature tuples represented in both signatures, and $meanRSSI(t)_r$ is the mean RSSI value in the signature tuple t appearing in signature r. Other distance metrics, such as Euclidean distance, can be used as well. In our experiments, the Manhattan and Euclidean distance metrics both produced very similar results, and the Manhattan distance is very efficient to compute on nodes with low computational capabilities.

Given the set of signature distances, the location of a mobile node can be calculated in several ways. The simplest approach is to take the centroid of the geographic location of the k nearest (in terms of signature space) reference signatures. By weighting each reference signature's location with the signature distance, we bias the location estimate towards "nearby" reference signatures. While this method is simple, using a fixed value for k does not account for cases where the density of reference signatures is not uniform. For example, in a physical location where few reference signatures have been taken, using the k nearest reference signatures may lead to comparison with signatures that are very distant.

Instead, we consider the centroid of the set of signatures within some ratio of the nearest reference signature. Given a signature s, a set of reference signatures R, and the nearest signature $r^\star = \arg\min_{r \in R} M(r, s)$, we select all reference signatures $r \in R$

that satisfy

$$\frac{M(r, s)}{M(r^\star, s)} < c$$

for some constant c. The geographic centroid of the locations of this subset of reference signatures is then taken as the mobile node's position. We find that small values of c work well, generally between 1.1 to 1.2 (see Section 6.9). In this paper, we choose a specific, empirically-determined value for c. An interesting future research question is how this parameter can be determined automatically.

4 Making RF-based Localization Robust

In this section, we describe our approach to making RF location tracking robust to beacon node failure and signal perturbations. MoteTrack must ensure that there are *no single points of failure* and that the location estimation algorithm can *gracefully handle incomplete data and failed nodes*.

We address the first requirement by making our system completely decentralized. The location estimation protocol relies only on local data, local communication between nodes, and involves only currently operational nodes. The reference signature database is carefully replicated across beacon nodes, such that each beacon node stores a subset of the reference signatures that is carefully chosen to maximize location tracking accuracy.

We address the second requirement by using an adaptive algorithm for the signature distance metric that accounts for partial failures of the beacon node infrastructure. Each beacon node dynamically estimates the current fraction of locally failed beacon nodes and switches to a different distance metric to mitigate location errors caused by these failures.

4.1 Decentralized location estimation protocol

Given a mobile node's signature s and a set of nearby beacon nodes contained in s, the first question is how to compute the mobile node's location in a way that only relies upon local communication. We assume that each beacon node stores a *slice* of the reference signature database (which may be partially or wholly replicated on other nodes). Using Mica2 motes as the beacons, the limited storage capacity (128KB ROM and 4KB of RAM) implies that the entire database will not generally be replicated across all beacon nodes.

In MoteTrack, a mobile node first acquires its signature s by listening to beacon messages, and then broadcasts s, requesting that the infrastructure send it information on the mobile node's location. One or more of the beacon nodes then compute the signature distance between s and their slice of the reference signature database, and report either a set of reference signatures to the mobile node, or directly compute the mobile node's location. Each of these designs is discussed in turn below.

k beacon nodes send their reference signature slice

In this first design, the mobile node broadcasts a request for reference signatures and gathers the slices of the reference database from k nearby beacon nodes. The mobile node then computes its location using the received reference signatures. While this approach can be very accurate, it requires a great deal of communication overhead. An alternative is to limit the amount of data that is transferred by contacting only $n < k$ nearby beacon nodes, requesting that each one only send the m reference signatures that are closest (in terms of signature distance) to s. For example, the mobile node can query the n beacon nodes with the largest RSSI value in s.

k beacon nodes send their location estimate

An alternative to the previous design allows each of the k beacon nodes to compute its estimate of the mobile node's location using its own slice of the reference signature database. These k location estimates are then reported to the mobile node, which can compute the "centroid of the centroids" according to its RSSI to each beacon. The mobile node simply transmits its signature s and receives k location estimates.

While this version has reasonable communication overheads, our initial evaluations indicated that it does not produce very accurate location estimates. The problem is that for k greater than one or two, some of the beacon nodes are too far from the mobile node and therefore do not store a very relevant set of reference signatures. Since this design does not seem to perform well, we abandoned it for the design described in the next section.

Max-RSSI beacon node sends its location estimate

Our third and final design combines the advantages from the first two to obtain both low communication overhead and accurate location estimates. In this design, we assume that the most relevant (closest in signature space) reference signatures are stored on the beacon node with the strongest signal. The mobile node sends a request to the beacon node from which it received the strongest RSSI, and only that beacon node estimates the mobile node's location. As long as this beacon node stores an appropriate slice of the reference signature database, this should produce very accurate results. The communication cost is very low because only one reply is sent to the mobile node containing its location coordinates.

4.2   Distributing the reference signature database to beacon nodes

Using the decentralized protocol described above, beacon nodes estimate locations based on a partial slice of the entire reference signature database. Therefore it is crucial that the reference signatures are distributed in an "optimal" fashion. In addition, we wish to ensure that each reference signature is replicated across several beacon nodes in case of beacon node failures. We use two algorithms for database distribution, which we refer to as *greedy* and *balanced*.

Greedy distribution algorithm

The *greedy* algorithm has one parameter: `maxRefSigs`, which specifies the maximum number of reference signatures that each beacon node is willing to store locally. The algorithm operates by iteratively assigning reference signatures to beacon nodes as follows. For each signature, a given beacon node accepts and stores the signature if (1) it is currently storing fewer than `maxRefSigs` or if (2) the new reference signature contains a greater RSSI value for the beacon node in question.

The advantages of the greedy approach are simplicity and no requirement for global knowledge or coordination between nodes. For example, beacon nodes can be updated individually without affecting the signatures stored on other beacon nodes.

Balanced distribution algorithm

One of the problems with the greedy algorithm is that some reference signatures may never get assigned to a beacon node, while others may be replicated many times. The *balanced* algorithm tries to strike a balance between pairing each beacon node with its closest reference signature, while evenly distributing reference signatures across beacons. This is a variant of a stable marriage algorithm. To ensure that no reference signature is paired with too many beacon nodes, the algorithm prevents the match if either

the current reference signature or beacon node have been assigned two more times than any other reference signature or beacon node.

The advantage of the balanced algorithm is that it can ensure balanced distribution of reference signatures while attempting to assign reference signatures to their closest beacon nodes. The disadvantage is that it requires global knowledge of all reference signature and beacon node pairings, and is therefore only appropriate for an offline, centralized initialization phase. If one wishes to update a small set of the beacon nodes, a complete reassignment involving all nodes and reference signatures may have to take place.

The pseudocode for both algorithms can be found in the technical report [18].

4.3 Adaptive signature distance metric

Given that we do not expect the set of signature tuples represented in the reference signature r and mobile node's signature s to be identical, there is a question about how to account for missing data in one signature or the other. If r contains a signature tuple not found in s, this can be due to s being taken at a different location in the building, or the failure of a beacon node. Taking the intersection of the beacon set in r and s is not appropriate, because we wish to capture the low intersection in cases where one signature is largely dissimilar to another.

First, we consider the case with no beacon node failures. In this instance, missing tuples between two signatures indicates that they are at different locations. We define the *bidirectional* signature distance metric as:

$$M_{bidirectional}(r, s) = M(r, s) + \beta \sum_{t \in (s-r)} meanRSSI(t)_s + \beta \sum_{t \in (r-s)} meanRSSI(t)_r$$

That is, each RSSI tuple not found in $(r \cup s)$ adds a penalty to the distance that is proportional to that signature's RSSI value. We determined empirically that value between 0.95 and 1.0 work well for β.

This distance metric is appropriate when few beacon nodes have failed, since it penalizes for all RSSI tuples not found in common between r and s. In case of beacon node failures, however, a larger number of RSSI tuples will appear in the set $(r - s)$, leading to an explosion of error. To minimize the errors introduced from failed nodes, we define the *unidirectional* distance metric:

$$M_{unidirectional}(r, s) = M(r, s) + \beta \sum_{t \in (s-r)} meanRSSI(t)_s$$

which only penalizes tuples found in s (the mobile node's signature) and not in r (a reference signature). Assuming that the reference signatures were acquired while all beacon nodes are operational, the unidirectional metric only compares signatures between operational nodes.

As an example, consider the following signatures:

r	s	
BN 1, RSSI 20	BN 1, RSSI 45	$M_{bidirectional} = \|20 - 45\| + \|90 - 60\|$
	BN 2, RSSI 15	$+15 + 70$
BN 3, RSSI 70		$M_{unidirectional} = \|20 - 45\| + \|90 - 60\|$
BN 4, RSSI 90	BN 4, RSSI 60	$+15$

For simplicity, we do not show multiple power levels in this example. As we will see in Section 6.9, when few beacon nodes have failed, the bidirectional distance metric achieves greater accuracy than the unidirectional metric, because its comparison space

is larger. With the unidirectional metric, only operational beacons are considered, but overall accuracy is diminished when few beacon nodes have failed.

Therefore, we employ an *adaptive* scheme that dynamically switches between the unidirectional and bidirectional metrics based on the fraction of local beacon nodes that have failed. Beacon nodes periodically measure their local neighborhood, defined as the set of other beacon nodes that they can hear. This neighborhood is compared to the *original neighborhood* (measured shortly after the system has been installed or reconfigured). If the intersection between the current and original neighborhoods is large, the bidirectional distance metric is used, achieving higher accuracy. If the fraction of failed nodes exceeds some threshold, the unidirectional distance metric is used instead.

This approach makes two assumptions. The first assumption is that the connectivity between beacon nodes does not change substantially over time. To mitigate this problem, we only include a beacon node in the original neighborhood if its RSSI is above some threshold. However, for the current neighborhood we include all beacon nodes regardless of RSSI, and that exist in the original neighborhood. Note that we only include a beacon node if it exists in the original neighborhood. This will eliminate cases when a beacon node's signal temporarily reaches more nodes. The second assumption is that there are no beacon node failures between the time that the reference signature database is collected and the system is deployed for normal operation. We believe this is a valid assumption for most installations and can be readily addressed by reinitializing the original neighborhood set of each node.

5 Implementation and Data Collection

MoteTrack is implemented on the Mica2 mote platform using the TinyOS operating system [19]. We chose this platform because it is designed for low-power operation, is relatively small, and can be deployed unobtrusively in an indoor environment. In addition, the motes incorporate a low-power 433/916 MHz FSK radio, the Chipcon CC1000, which provides both programmable transmission power levels and direct sampling of received signal strength. We expect that MoteTrack could be readily ported to use forthcoming 2.4 GHz 802.15.4 radio chips. We note that MoteTrack runs entirely on the mote devices themselves and does not require a supporting infrastructure, such as back-end servers or PCs, in order to operate. A laptop connected to a mote is used to build the reference signature database, but thereafter the system is self-contained.

The total code size for the beacon and mobile node software is about 3,000 lines of NesC code. In our implementation, the reference signatures for each beacon node are loaded into program memory on the mote storing that segment of the database. This could be readily modified to use a combination of RAM and serial flash or EEPROM. Recall that each beacon node stores a different set of reference signatures depending on the distribution mechanism used.

5.1 Deployment

We have deployed MoteTrack over one floor of our Computer Science building, measuring roughly $1742\,m^2$, with $412\,m^2$ of hallway area and $1330\,m^2$ of in-room area. Our current installation consists of 20 beacon motes (Figure 2).

We collected a total of 482 reference signatures. Each signature was collected for 1 minute, during which time every beacon node transmitted at a rate of 4 Hz, each cycling through 7 transmission power levels (from -20 to 10 dBm in steps of 5 dBm).

We note that in a normal deployment, a *much* smaller dataset is required and the amount of time spent collecting a signature can be on the order of several seconds rather than 1 minute. The large number of reference signatures was gathered in order to evaluate the system under various conditions and parameters. Likewise, we collected many
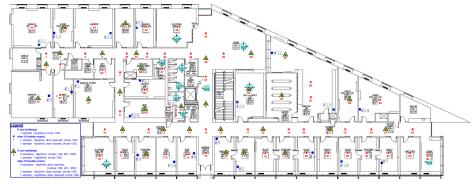
Fig. 2. Signature locations in the testing data set. *The blue dots represent the fixed beacon nodes. The red squares represent acquired signature locations; those with a green triangle were tested with 3 different motes.*

	Training data	Testing data
Total signatures	282	200
Daytime	282	170
Nighttime	–	30
Using 3 motes	–	30
Hallway	151	79
In room, door open	67	81
In room, door closed	64	40

Fig. 3. Summary of the number of samples for each scenario of the training and testing data.

samples for each beacon message transmission power pair, because we suspected the RSSI to vary across samples; however, we discovered that there is *very* little variation between samples and therefore we only need on the order of 2 to 3 samples.

A *beacon message* consists of a three-byte payload: 2 bytes for the source node ID and one byte representing the transmission power level. Therefore, all beacon messages from a source node ID require $2+T$ bytes: 2 bytes for the ID and $T*1$ bytes for each of the T power levels, i.e. $\{sourceID, RSSI_{p=1}, ..., RSSI_{p=T}\}$. A complete *reference signature* consists of 6 bytes for the location size (3 coordinates time 2 bytes per coordinate), 2 bytes for the ID, and up to N beacon nodes with T power levels each. The storage overhead for one reference signature is therefor $6+2+N*(2+T) = 8+2N+TN$ bytes. In our deployment, $N = 20, T = 7$, for a total of 188 bytes per reference signature. The code size for MoteTrack is about 20 KB, leaving 108 KB of read-only SRAM on each beacon node for storing a partition of the reference signature database. Therefore each beacon node can store up to 588 reference signatures. In Section 6.3 we discuss the impact of limiting the amount of per-beacon storage to estimate the effect of much larger reference signature databases.

We divided the collected signatures into two groups: the *training data set* (used to construct the reference signature database) and the *testing data* (used only for testing the accuracy of location tracking). Our analysis investigates effects of a wide range of parameters, including whether signatures are collected in a hallway or in a room, whether the room's door is open or closed, the time of day (to account for solar radiation and building occupancy) and the use of different mobile nodes (to account for manufacturing differences). We collected at least 30 signatures for each of the various parameters to ensure that results are statistically significant. Figure 2 shows a map of the testing data sets, and Figure 3 summarizes the data.

6 Evaluation

In this section we present a detailed evaluation of the performance of MoteTrack along a number of axes. First, we look at the overall accuracy of MoteTrack. Although accuracy is not our focus, we do need to understand how the system performs under various parameters. We evaluate the accuracy on our entire floor which includes hallways and rooms, the location estimation protocols, algorithms for selecting reference signatures, type of database distribution, number of transmission powers used, and the density of beacon nodes and reference signatures.

Second, we look at robustness with no beacon node failures. Here we investigate the effects of radio signature perturbations, using different motes, time of day, and obstacles such as doors.

Finally, we look at robustness with beacon node failures. Here we examine how MoteTrack performs under extreme failures of the beacon infrastructure and evaluate our adaptive signature distance metric.

These results were obtained using an offline simulation of the MoteTrack protocol in order to give us the maximum flexibility in varying experimental parameters. In all cases the real reference signature database acquired in our building was used to drive the simulation. The simulator captures the effect of beacon node failure, RF perturbations, distribution of the reference signature database, and the different algorithms for signal distance and centroid calculation. The system is fully implemented on real motes and we have demonstrated a full deployment of MoteTrack in our building along with a real-time display of multiple user locations superimposed on a map.

6.1 Location estimation protocols

We first evaluate the accuracy of the system over the entire floor in the context of three location estimation protocols. Two decentralized location estimation protocols and a centralized one: having a closest (in terms of RSSI) beacon node compute the location, receiving reference signatures from several ($k = 3$) nearby beacon nodes, and computing the location based on all of the received signatures. The centralized version is used as a benchmark for comparison purposes.
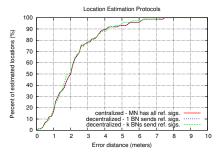
Figure 4 shows the cumulative distribution function (CDF) for the protocols. As we can see, the accuracy of the 3 versions is nearly identical suggesting that the closest beacon node does in fact store most of the relevant reference signatures for accurately estimating the mobile node's location. Likewise, the additional overhead of receiving reference signatures from k beacon nodes is unjustified.

Our deployment uses the first decentralized protocol (i.e., closest or maxRSSI beacon node sends the location estimate), and it's the accuracy a user of the system should expect to get. As we can see, 50% and 80% of the location estimates are within 2 m and 3 m respectively from their true location. This is more than adequate for applications that require locating persons, such as tracking the location of rescue personnel or locating patients.

For the rest of this section we consider only the decentralized version where the closest beacon node computes the location.

6.2 Selection of reference signatures

The next parameter of interest is the algorithm used to select reference signatures that are close (in terms of signal space) to the mobile node's signature. Figure 5 compares the k-nearest selection approach to the relative signature distance threshold technique. The k-nearest algorithm computes the centroid location of the k closest reference signatures. The relative threshold scheme limits the set of reference signatures based on a threshold that is proportional to the signature distance to the nearest reference signature.
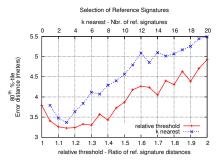
Fig. 4. Location Estimation Protocols *Under normal circumstances both decentralized protocols perform nearly identical to the centralized one.*

Fig. 5. Two reference signature selection algorithms. *The relative threshold algorithm performs better than the k-nearest one.*

For k-nearest, small values of k are appropriate for computing the location centroid, but values above this introduce significant errors. The relative thresholding scheme is more accurate as it limits the set of locations considered according to the signature distance metric. The optimal distance threshold is around 15-20% of the closest reference signature.

6.3 Distribution of the reference signature database

Next we look at the different techniques for replicating reference signatures across beacon nodes. This aspect of the design is crucial because each beacon node stores only a subset of the full signature database. We look at two algorithms: *greedy* and *balance* reference signature distribution. For each of these we also vary whether a given signature is stored only on the beacon node that is closest to the signature (*closest BN*), or replicated across $k = 3$ beacon nodes (*k=3 BN*). To estimate the effect of growing the reference signature database beyond its current size (282 signatures), we artificially limited the maximum number of reference signatures that each beacon node could store.
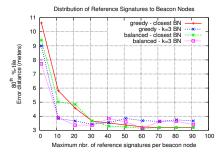
Figure 6 shows the results of this experiment. As the maximum storage capacity of each beacon node is decreased, the balanced distribution achieves the best results. In most cases, replicating each signature across k beacon nodes achieves better results than storing it only on the closest beacon node. When the memory capacity of the beacon nodes is not limited, there is less noticeable difference between the approaches as it is more likely that any given beacon node has the relevant set of signatures.

6.4 Transmission of beacons at multiple power levels

Recall that beacon nodes cycle through transmitting beacons at different power levels ranging from -20 dBm to 10 dBm. Initially it was not clear if transmitting at multiple power levels would noticeably improve accuracy. Figure 7 shows the 80th percentile error distance as the number of beacon transmission power levels is varied. The error is averaged across all combinations of N power levels, with N ranging from 1 to 7, i.e. $\binom{7}{N}$. As the figure shows, increasing the diversity of power levels increases the 80th percentile accuracy by nearly 2 m. However, increasing the number of transmission power levels involves a trade-off in terms of higher storage for reference signatures.

6.5 Density of beacon nodes and reference signatures

Of particular interest to someone deploying MoteTrack is the number of beacon nodes and reference signatures needed to achieve a certain accuracy. For this experiment we
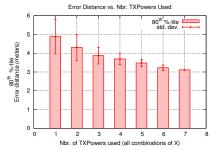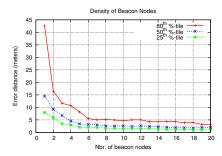
Fig. 6. Greedy vs. balanced distribution of the reference signature database. *When the memory size of beacon nodes is limited, the balanced algorithm outperforms the greedy one.*

Fig. 7. The effect of varying the number of transmission power levels used to transmit beacons. *Increasing the diversity of beacon power levels increases accuracy considerably.*
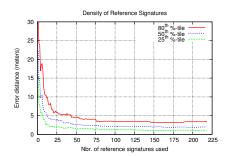

Fig. 8. Density of beacon nodes. *After 6 to 7 beacon nodes ($0.004 \frac{beacon\ nodes}{m^2}$) additional beacon nodes provide diminishing returns.*

Fig. 9. Density of reference signatures. *The accuracy begins to stabilize after 75 reference signatures ($0.043 \frac{reference\ signatures}{m^2}$).*

artificially restricted the set of beacon nodes represented in the reference signature database. For each number of beacon nodes we hand-selected the appropriate number of nodes that were approximately uniformly distributed throughout the building, avoiding any "clusters" of beacon nodes.
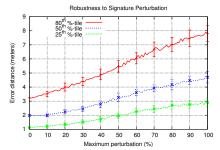
Figure 8 shows how location error varies with the number of beacon nodes deployed in the building, which also represents the overall density of nodes. It appears that there is a critical number of beacon nodes required after which the accuracy of the system increases marginally. In this case the critical density is around 6 to 7 nodes which is about $0.004 \frac{beacon\ nodes}{m^2}$.

Likewise, varying the number of reference signatures has a strong effect on location tracking accuracy. Figure 9 shows that the error distance decreases quickly up to the first 25 reference signatures and begins to stabilize after 75 reference signatures, representing a signature density of $0.043 \frac{reference\ signatures}{m^2}$.

6.6 Robustness to perturbed signatures

We now turn our attention to the robustness of the system under no beacon failures. We begin by looking at the effects of radio signature perturbations.

The RF propagation in a building may change slightly over time or more drastically in a disaster, when the building's characteristics may alter from events such as walls col-
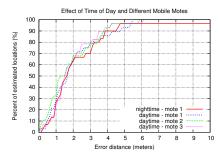
Fig. 10. Robustness to the perturbation of signatures' RSSI measurements. *The accuracy of MoteTrack degrades linearly with increased perturbation levels. All results are averaged over 30 trials.*

Fig. 11. Effects of time of day and manufacturing differences between motes. *No consistent differences are found when varying the time of day or the mobile node.*

lapsing. To understand these implications, we evaluate how the accuracy of MoteTrack changes for various perturbation levels of a signature's RSSI measurements.

For each percentage, we perturbed the RSSI measurements of all signatures (i.e. the testing data) by up to a *maximum percentage* of the entire RSSI range. The perturbation amount for each RSSI is taken from a uniform distribution between zero and maximum percentage. As we can see in Figure 10, MoteTrack is quite robust to RSSI perturbations. For a maximum perturbation of 40%, the 80th percentile has an accuracy of under 5 m and, for the 50th and 25th percentiles it has an accuracy of under 3 m and under 2 m respectively.

6.7 Time of day and different motes

Next we look at the effects of two other parameters: the time of day and manufacturing differences between motes. Time of day examines how the system reacts to changes in building occupancy and movement; the use of different motes accounts for the overall effect on the system from variation between motes.
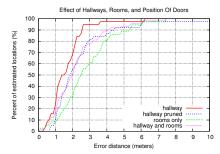
For this experiment we collected a daytime data set between the hours of 9:00am and 4:00pm on a weekday, using 3 different motes. We also collected a nighttime data set at 1:00am, when few occupants are in the building, using only a single mote. Mote 1 was used to collect a larger number of data points. To ensure a fair comparison, only the locations that are common to all four data sets (three motes during the day and one mote at night) were used here.

Figure 11 shows that the accuracy is largely unaffected by these parameters, so we expect it to work well even for different mobile nodes and times of day.

6.8 Effect of hallways, rooms, and door position

Hallways tend to act as waveguides while walls and doors contribute to signal attenuation. We first investigate how the accuracy of a mobile user is affected by its location in the building. Figure 12 shows the cumulative distribution function (CDF) of the location error for signatures obtained in the hallway and inside rooms, with doors opened and closed. In the hallway, nearly 80% of location estimates are within 2 m of their true location, while in rooms the 80th percentile is slightly under 4 m.

As we can see from the table in Figure 3, the density of reference signatures in the hallway is higher then inside rooms. In the hallway the density is $0.36 \frac{\text{reference signatures}}{m^2}$
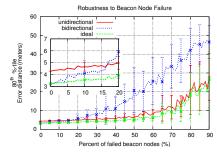
Fig. 12. The effects of hallways, rooms, and position of doors on location tracking accuracy. *This data represents a limited data set in which the density of reference signatures in the hallway was pruned to match that of the in-room reference signatures.*

Fig. 13. Robustness to beacon node failure. *The unidirectional algorithm is more robust to large beacon node failures, but yields poorer accuracy when there are fewer failures. All results are averaged over 30 trials. The vertical bars represent the standard deviation.*

and inside rooms it is $0.05 \frac{\text{reference signatures}}{m^2}$. In order to make a fair comparison, we pruned the hallway data set to have the same density as the rooms data set, and plotted the pruned hallway data set (labeled *hallway pruned*). As we can see, the error distance for the 80th percentile increased to just under 2.9 m. For *hallway and rooms*, after pruning the hallway part of the data set, we found a slight increase in error from 3.2 m to 3.5 m. We also looked at the effect of the position of doors and found that they don't make a significant difference.

6.9   Robustness to beacon node failure

Finally, we evaluate MoteTrack's ability to continue providing accurate location estimates even when a large number of beacon nodes have failed. We consider this aspect of MoteTrack to be essential for its potential use in disaster response scenarios. Here, we simulate the effect of failed beacon nodes by selectively eliminating beacon nodes from mobile node signatures, as well as preventing those beacon nodes from participating in the decentralized location calculations.

We evaluated robustness to failure using both the unidirectional and bidirectional algorithms for calculating signature differences. Beyond a certain failure threshold, we expect the unidirectional version to perform better than the bidirectional version, since it only considers RSSI values from beacon nodes that are present at the time the signature is constructed. As we can see in Figure 13, after about 16% of the beacon nodes have failed, the unidirectional version indeed produces more accurate results.

For comparison purposes, we also show the ideal case. The ideal case is when we have perfect knowledge of which nodes failed. This is the best case scenario and although under normal circumstances it's unachievable in a completely decentralized system, it shows the lower bound. In this case the bidirectional algorithm is used but only over RSSI measurements from nodes that did not fail.

Although unidirectional signal distance is more robust, it is less accurate when there are few failed nodes. As mentioned in Section 4.3, MoteTrack decides dynamically which algorithm to use based on the local failure percentage that it last computed. MoteTrack starts out using the bidirectional algorithm and after it estimates that the beacon failure is greater than 16%, it switches to the unidirectional algorithm.

7   Future Work

The current system as described in the paper, requires an offline installation and calibration prior to use. In many cases, such as responding to a mass casualty incidents in an arbitrary area (e.g. a train wreck or a multi-car highway accident), pre-installation and calibration of a beacon node infrastructure is clearly not feasible. For these scenarios, we need an *ad hoc* mechanism for rapidly deploying the location tracking system and populating the beacon nodes with reference signatures.

In an outdoor environment, one approach is to leverage GPS to automatically populate the signature database. For example, medics responding to the scene of a disaster can place beacon nodes at well-spaced (and arbitrary) points around the site. Rather than require every patient or medic to carry a GPS receiver (which are often higher power and bulkier than sensor motes), several medics can carry a PDA equipped with a GPS receiver and MoteTrack transceiver. The PDA can automatically record reference signatures as the medics move around the site, populating the reference signature database on the fly using the greedy distribution, which does not require global knowledge of the beacon nodes and reference signatures. Signature acquisition can be performed rapidly, since each signature requires only a few beacon messages from each node and transmission power, which can be acquired in a very short period of time [20]. In our experiments we obtained good results in about one second. Location tracking accuracy will improve over time as more reference signatures are acquired.

One of the challenges faced is how to deal with the additional error introduced by the GPS location estimate. While in North America, GPS devices using the Wide Area Augmentation System (WAAS) can yield location estimates to within 3 m 95% of the time [21], it is not clear how much this will impact the overall accuracy of the system.

We have already ported our system to the Telos and MicaZ mote platforms, which support the CC2420 802.15.4 radio chip, and observed very similar results in terms of accuracy. For the immediate future, we intend to re-run our measurements with identically-placed beacon nodes to directly compare the performance of MoteTrack with 433 MHz and 2.4 GHz radios.

8   Conclusions

In this paper, we describe how to extend the basic RF approach for localization in order to make it highly robust and decentralized. We achieve this through a decentralized location estimation protocol that relies only on local data, local communication, and operational nodes; by replicating the reference signature database across beacon nodes in a fashion that minimizes per-node storage but achieves high level of robustness to failure; and by using a dynamic signature distance metric that handles incomplete data and adapts to the locally failed beacon nodes.

We implemented, deployed, and extensively evaluated our approach through a system called MoteTrack, based on the Berkeley Mica2 mote. We choose this platform because we believe that many of the applications where robustness is important will want to use small, inexpensive devices that can be embedded in the environment such as walls, in the equipment of rescue personnel, or integrated with vital-sign sensors placed on patients [22].

MoteTrack achieves a 50th and 80th percentile of 2 and 3 meters respectively, and can tolerate a failure of up to 60% of the beacon nodes and signature perturbations of up to 50%, with negligable increase in error.

References

1. Bahl, P., Padmanabhan, V.N.: RADAR: An In-Building RF-Based User Location and Tracking System. In: INFOCOM. (2000) 775–784

2. Youssef, M., Agrawala, A., Shankar, A.U.: WLAN location determination via clustering and probability distributions. In: IEEE PerCom 2003. (2003)

3. Slack, G.: Smart Helmets Could Bring Firefighters Back Alive. FOREFRONT (2003) Engineering Public Affairs Office, Berkeley.

4. McKinsey & Company: Increasing fdny's preparedness. http://www.nyc.gov/html/fdny/html/mck_report/index.shtml (2002)

5. Myllymaki, P., Roos, T., Tirri, H., Misikangas, P., Sievanen, J.: A Probabilistic Approach to WLAN User Location Estimation. In: Proc. The Third IEEE Workshop on Wireless LANs. (2001)

6. Smailagic, A., Small, J., Siewiorek, D.P.: Determining User Location For Context Aware Computing Through the Use of a Wireless LAN infrastructure (2000)

7. Ray, S., Starobinski, D., Trachtenberg, A., Ungrangsi, R.: Robust Location Detection with Sensor Networks. IEEE JSAC **22** (2004)

8. Krumm, J., Williams, L., Smith, G.: SmartMoveX on a Graph – An Inexpensive Active Badge Tracker. In: UbiComp 2002. (2002)

9. Bahl, P., Balachandran, A., Padmanabhan, V.: Enhancements to the RADAR User Location and Tracking System. Technical Report 2000-12, MSR (2000)

10. Castro, P., Chiu, P., Kremenek, T., Muntz, R.R.: A Probabilistic Room Location Service for Wireless Networked Environments. In: UbiComp. (2001)

11. Pandya, D., Jain, R., Lupu, E.: Indoor Location Estimation Using Multiple Wireless Technologies. In: IEEE PIMRC. (2003)

12. Christ, T., Godwin, P.: A Prison Guard Duress Alarm Location System. In: IEEE ICCST. (1993)

13. Hightower, J., Want, R., Borriello, G.: SpotON: An Indoor 3D Location Sensing Technology Based on RF Signal Strength. Technical Report UW CSE 00-02-02, University of Washington (2000)

14. Priyantha, N.B., Chakraborty, A., Balakrishnan, H.: The Cricket Location-Support System. In: MobiCom. (2000)

15. Priyantha, N.B., Miu, A., Balakrishnan, H., Teller, S.: The Cricket Compass for Context-Aware Mobile Applications. In: Proc. 7th ACM MobiCom. (2001)

16. Ward, A., Jones, A., Hopper, A.: A New Location Technique for the Active Office. IEEE Personal Comm. **4** (1997)

17. Want, R., Hopper, A., Falcao, V., Gibbons, J.: The Active Badge Location System. Technical Report 92.1, Olivetti Research Ltd. (ORL) (1992)

18. Lorincz, K., Welsh, M.: A Robust, Decentralized Approach to RF-Based Location Tracking. Technical Report TR-19-04, Harvard University (2004)

19. Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D.E., Pister, K.S.J.: System Architecture Directions for Networked Sensors. In: ASPLOS 2000. (2000) 93–104

20. Krumm, J., Platt, J.: Minimizing Calibration Effort for an Indoor 802.11 Device Location Measurement System. Technical Report MSR-TR-2003-82, Microsoft Research (2003)

21. WAAS: What is WAAS? (2005) http://www.garmin.com/aboutGPS/waas.html.

22. Lorincz, K., Malan, D.J., Fulford-Jones, T.R.F., Nawoj, A., Clavel, A., Shnayder, V., Mainland, G., Moulton, S., Welsh, M.: Sensor Networks for Emergency Response: Challenges and Opportunities. IEEE Pervasive Computing (2004)