

Sensor Field Localization: A Deployment and Empirical Analysis

Kamin Whitehouse, Fred Jiang, Chris Karlof, Alec Woo, David Culler
{kamin,xjiang,ckarlof,awoo,culler@cs.berkeley.edu}
Computer Science, UC Berkeley

UC Berkeley Technical Report UCB//CSD-04-1349
April 9, 2004

Abstract

This paper presents a deployment methodology for sensor field localization systems. To deploy our ultrasound based localization system, we first create a simulation environment that captures real-world ranging characteristics. The localization system can then be designed against the ranging noise of a particular environment and ranging implementation. The simulation environment can also be used to estimate the localization accuracy and probability of successful localization before the deployment actually takes place. We evaluate this methodology through the empirical deployment of a 49 node, 8 hop network in which the system achieves an average accuracy of about 50cm.

1 Introduction

This paper presents a *sensor field localization* deployment, in which nodes in an ad-hoc wireless sensor network locate themselves with respect to a fixed coordinate system. Most localization systems such as GPS and Cricket [6, 11, 2, 4] use *single-hop* localization in which each node must be directly connected to at least several *anchor nodes* with known locations. In a sensor field, however, most nodes are not in contact with any anchor nodes. Sensor fields must use *multi-hop* localization in which each node obtains information about anchor nodes through its neighbors. Designing algorithms for multi-hop localization is an active area of research.

In this paper, we do not present a new localization algorithm. Instead, we use an existing algorithm and explore the process of taking it through the stages of hardware design, software implementation, and final deployment. We provide an empirical evaluation of many ideas in the literature, serving as a proof-of-concept in some cases and revealing tacit assumptions or unexpected problems in others.

To estimate the distance between nodes we use ultrasonic ranging for high accuracy and a conical reflector to achieve omni-directionality in a plane. To perform localization, we created a

completely distributed implementation of the Ad-hoc Positioning System (APS) DV-distance algorithm. In this paper we describe both implementations.

The principle contribution of the paper, however, is not the implementation of the system but the iterative methodology we use to deploy it. The framework of our deployment methodology is a realistic simulation environment. We found the typical simulation environment for localization to be very different from the real deployment environment. We designed a data collection technique that captures a complete profile of the physical environment and the particular ranging implementation. These profiles are then used to create simulation environments that accurately capture real-world ranging characteristics.

Given a realistic simulation environment, we can design the system against the particular noise characteristics of our environment and ranging implementation. In our case, outliers and ranging loss were identified as dominant sources of error in our environment that are not captured in traditional ranging models. In simulation, we could specially design a filter for the outlier distribution and implement what we call *range sharing* to ameliorate the effects of loss. The realistic simulation environment can be used to iteratively evaluate and redesign these techniques.

After the system is designed, our realistic simulation environment is used again to estimate the probability of successful localization in a given deployment scenario. This pre-deployment verification step can help identify, for example, the correct network density and number of anchor nodes that should be used without requiring days of trial and error in the field. These deployment parameters can be tuned to balance deployment costs with localization accuracy and probability of successful localization. This will be increasingly important for large or mission critical deployments that can only be deployed once.

In this paper we follow this methodology through an actual deployment and evaluate where it successfully improved the deployment process and where it did not. The deployment itself is

also compared with our simulation environment to evaluate how well the simulations predict real-world behavior of a localization system. The final deployment was a 49 node network deployed over a 144 square meter (1600 square foot) area and localized all nodes with a median error of .53m (1.7 feet).

In Section 2 we describe the hardware and software implementation. In Section 3 we describe the process of collecting empirical profiles of the environment and ranging implementation and using them for simulation. In Section 4 we describe the process of designing the system against specific environmental sources of error. In Section 5 we describe pre-deployment verification in which we estimate localization error the likelihood of successful localization before deployment. In Section 6 we detail unexpected problems encountered during deployment. In Sections 7 and 8 we present the results and analysis of the final deployment.

2 System Implementation

The implementation of this system builds upon and improves some of the best hardware designs and algorithms from several existing systems to create a unified system that is specially tailored to this localization problem. In this section, we detail our implementation and explain our motivations for each design decision. The focus of this paper, however, is neither the ranging and localization techniques used nor their implementations but rather the methodology presented in following sections that we use to actually bring these techniques to deployment.

2.1 Ultrasonic Ranging

Our ultrasound hardware combines and improves ideas from several ultrasound implementations. Our ultrasonic transducer circuitry is derived from that of the Medusa node [14], except that we add a switchable circuit so that a single transducer could be used to both transmit and receive. Our nodes measure ultrasonic time of flight by transmitting the acoustic pulse simultaneously with a radio message so that receivers can measure the time difference on arrival (TDOA) as described in Cricket [11]. When the transducers are face to face, our implementation can achieve up to 12m range with less than 5cm error. Comparable implementations were able to achieve proportionally similar results of 3-5m range with 1-2cm accuracy [14, 13, 8]. The differences in magnitude are due in part to our design decision to reduce the center frequency of the transducer from the standard 40kHz to just above audible range at 25kHz, which increases both maximum range and error.

Ultrasound transducers are highly directional, and small variations from a direct face to face orientation can have large effects on error and con-

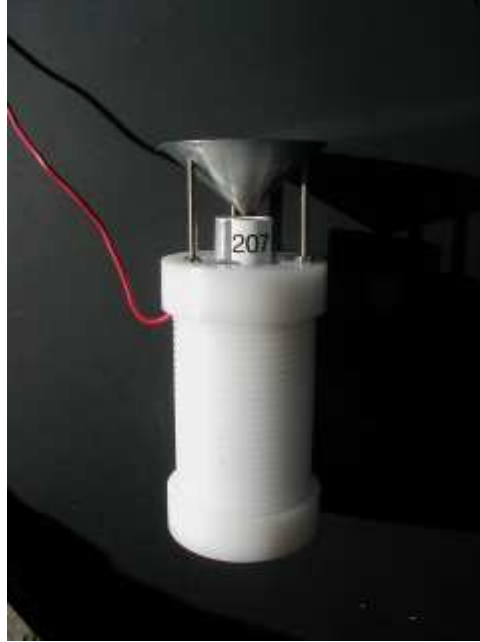


Figure 1: **Sensor Node** The white enclosure contains a Mica2Dot and battery and supports a reflective cone above the ultrasonic transducer that protrudes from the top.

nectivity. Two solutions have been proposed to use ultrasound in multi-hop networks: aligning multiple transducers outward in a radial fashion [13] or by using a metal cone to spread and collect the acoustic energy uniformly in the plane of the other sensor nodes [8]. We implemented the latter solution as shown in Figure 1. In this configuration, our nodes achieve about 5m range and 90% of the errors are within 6.5cm. A comparable implementation achieved about 3m range [8].

The ultrasound transducer is connected to an Atmel Atmega8 1MHz microcontroller which is used for both transmitting and receiving ultrasound signals. The output of the transducer is wired to the analog comparator on the microcontroller for detecting incoming signals through simple threshold detection which can be controlled in software through a digital potentiometer. The input of the transducer is wired to a PW line on the Atmega8, which directly keys the 25kHz signal. Both the transducer and the microcontroller are mounted as a separate board attached to the Mica2Dot [5], which consists of a ChipCon CC1000 FSK 433Mhz radio and an Atmel Atmega128 4MHz microcontroller. Because the radio and ultrasonic transducer are controlled by different microcontrollers, a single interrupt line is used for precise time synchronization between them. The two microcontrollers exchange timing and ranging information through an I2C communication bus. The schematics for this de-

sign are freely available at [1].

2.2 Localization

For localization we used the Ad-hoc Positioning System’s (APS) DV-distance algorithm [9], which is representative of a large class of distributed localization algorithms that use shortest-path [16, 12, 20] or bounding-box [19, 15] approximations. APS uses a distance vector algorithm to approximate the shortest path distance through the multihop network from each node to each of the anchor nodes. Each shortest path distance approximates the true distance to the anchor, reducing the multi-hop localization problem to a single-hop localization problem with a more complex range estimate. The approximate distance to each anchor is then used with the anchor node positions to triangulate the position of each node using linear least-squares.

APS has been shown to yield comparable results to the other distributed localization algorithms [7] and, intuitively, all of these algorithms suffer from the same two sources of error: they will overestimate distances in sparse networks and underestimate distances in the face of large ranging noise.

In our implementation, the APS algorithm runs in three fully decentralized phases. When the anchor nodes are given their positions, they trigger a *ranging phase* in which all nodes estimate the distance to each of their direct ranging neighbors. The anchors then initiate a *shortest path phase*, in which anchors initiate a tree broadcast, allowing each node to determine its shortest path to each anchor in a distance vector manner. When all broadcasts are complete, each node estimates its position in the *localization phase*. After the anchor nodes were given their positions, the entire process was automated with no human intervention or central computer and completed in less than five minutes for each deployment. All ranging estimates, shortest paths and estimated locations were stored in RAM on the nodes and were collected by an automated script after each run.

3 Generating Realistic Simulations

In traditional simulation, data is generated from a parametric function. For example, ranging noise is often modeled as Gaussian noise with the function $\mathcal{N}(d_{ij}, \sigma)$ and connectivity is modeled as a *Unit Disk* using the inequality $d_{ij} \leq d_{max}$. For traditional simulation to be meaningful, the model parameters d_{max} and σ must be estimated from empirical ranging data. The typical data collection technique for ranging is to place a transmitter and receiver at several known distances and measure the response [18, 10, 13].

However, we found that this commonly used simulation technique yields optimistic predictions of how well the localization system will actually perform; we have observed true deploy-

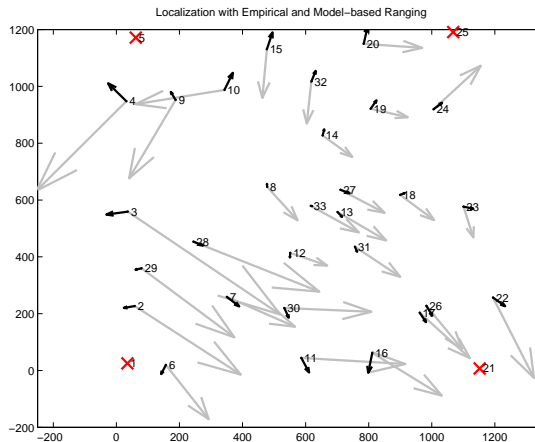


Figure 2: **Error Model Evaluation** The black dots with numbers are the true node positions and the “X”s are the anchor nodes. The dark arrows show the localization error vectors when using simulated ranging data with 5m maximum range and 20cm Gaussian noise. The light arrows show the errors when the same algorithm is executed using empirical data with similar characteristics.

ments to yield accuracy four to five times worse than predicted by simulation. Figure 2 compares the errors in simulation resulting from data generated by parametric functions versus empirical data, using the same algorithm on the same topology. Nodes could localize with an average error of only about 35cm with parameterized ranging error but saw almost 300cm error with empirical data.

3.1 Statistical Sampling

We developed an alternative simulation technique based on *statistical sampling* where we generate data for simulation by randomly drawing measurements from an empirical data set. Define the distribution $M(\delta, \epsilon)$ to be the empirical distribution of all observed ranging estimates for distances in the interval $[\delta - \epsilon, \delta + \epsilon]$. We generate a ranging estimate \hat{d}_{ij} for simulation by using the error of a random sample from $M(d_{ij}, \epsilon)$. For example, if \hat{d} is the empirical estimate selected from $M(d_{ij}, \epsilon)$, then

$$\hat{d}_{ij} = d_{ij} + (\hat{d} - \hat{d}_a) \quad (1)$$

where \hat{d}_a is the actual distance at which \hat{d} was measured. Because $\hat{d} \sim M(d_{ij}, \epsilon)$, the simulation is using empirical distributions for signal noise and connectivity as long as $M(d_{ij}, \epsilon)$ accurately represents ranging characteristics at d_{ij} . The set $M(\delta, \epsilon)$ can include *ranging failures*, which are instances when a pair of nodes failed to obtain a ranging estimate. Ranging failures are

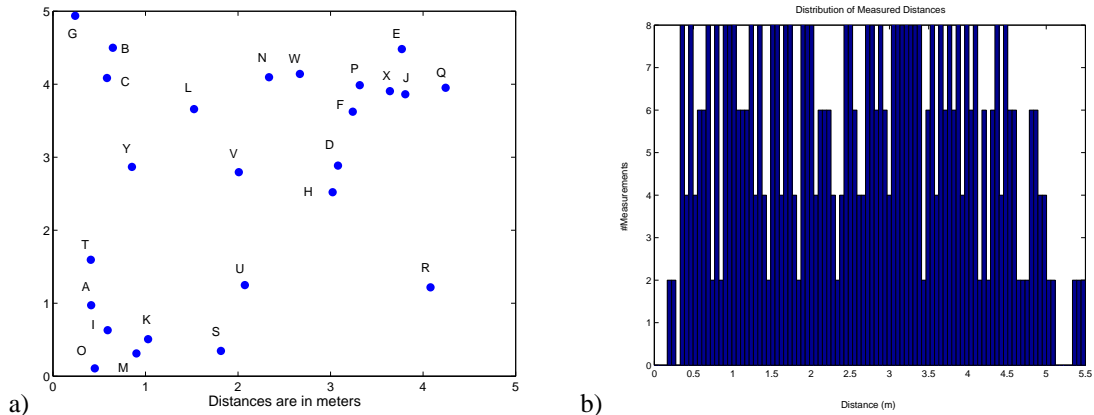


Figure 3: **Data Collection** a) This specially generated topology with 25 nodes measures 300 different distances with at least 1 distance every .025m between 0.4m and 5.2m. b) A histogram of the distances measured.

necessary to correctly model ranging connectivity.

3.2 Empirical Profiling

The challenge in using this sampling technique is to collect ranging error and connectivity data with a high enough resolution so that small values of ϵ can be used. For example, if we want to use $\epsilon = 2.5\text{cm}$ and ultrasound ranging has a maximum range of 10m, we must take empirical ultrasound measurements at 400 different distances. Instead of measuring each distance with a single pair of nodes, all measurements can be taken at once with $\sqrt{400} = 20$ nodes in a topology where each pair of nodes measures a different distance. By adding a few additional nodes, we can get multiple pairs at each distance. We generated such topologies using rejection sampling, i.e., we generated thousands of topologies until one of them exhibited the desired properties. For example, we used the topology in Figure 3.a, which required 25 nodes to obtain 2.5cm resolution over 5m, to characterize ultrasound. Figure 3.b shows the distribution of the distances that are measured by that topology.

All nodes are placed at random orientations in this topology and each node transmits 10 times in turn while all other nodes receive. To remove the bias of each distance being measured by only two pairs of nodes (the reciprocal pairs A/B and B/A), we repeated this procedure five times with different mappings of nodes to the topology locations. These mappings were generated using rejection sampling to ensure that the same distances were not always measured by the same pairs. The procedure generated 100 total measurements at each distance with 10 different transmitter/receiver pairs. Therefore, with $\epsilon = 0.05\text{m}$ (two inches) the set $M(\delta, \epsilon)$ is likely

to include 400 empirical measurements

Unlike the conventional pairwise data collection technique described above, the empirical measurements in $M(\delta, \epsilon)$ are taken with dozens of transmitter/receiver pairs, capturing a broad spectrum of node, antenna, and orientation variability. Furthermore, the measurements are taken over several different paths through the environment, capturing variability due to dips, bumps, rocks or other environmental factors. Finally, this technique captures connectivity characteristics by fixing the number of transmissions and measuring the number of readings at each distance. In contrast, the conventional pairwise technique described above requires the experimenter to take readings at every possible distance, hiding the degradation of ranging connectivity with distance.

Thus, the data generated by this data collection technique represents a complete empirical profile of the physical environment, the ranging implementation, and the interaction between the two. Because of the high resolution of the data, this empirical profile can be used to generate a very accurate simulation environment through the statistical sampling techniques described above. The rejection sampling algorithms required on average twelve hours to compute the topology and node mappings. Each data collection process required approximately 6 hours to complete, with the bulk of the time needed for data collection and to precisely measure out the special topology.

4 Designing for Real-world Errors

No ranging data will exhibit perfectly disk-like connectivity or gaussian noise. Given a simulation environment that can capture the subtleties of our environment and ranging implementation,

we can design our localization system against the ranging characteristics that fall outside of our parametric model of ranging. Furthermore, we can use simulation to design and evaluate techniques that are specifically tailored to our environment.

In the environments we tested, ranging characteristics differ in many ways from the theoretical ranging models but only a few will affect our localization algorithm. The APS localization algorithm in general is particularly sensitive to two conflicting forces. On one hand, the shortest path between a node and an anchor is almost never a straight line, and the zig-zag nature of these paths serves to lengthen them. On the other hand, the Bellman-ford algorithm selectively chooses range estimates with negative errors, so as the magnitude of errors increases the shortest path estimates become shorter. Whether the shortest path estimates underestimate or overestimate the true distances depends on the balance between the denseness of the connectivity graph and the amount of error in the ranging estimates.

Given this, the two aspects of our environmental noise that predominantly affect the APS algorithm are outliers and loss. *Outliers* are ranging estimates with error well outside the mean. This serves to shorten the shortest paths beyond that predicted by theoretical ranging models. *Loss* is the failure of two nodes closer than the maximum range of ultrasound to generate a distance estimate. This reduces connectivity and increases the zig-zag nature of the shortest paths.

In this section, we discuss the analysis of outliers and loss and the techniques we use to ameliorate their effects on the APS algorithm. Figure 4 depicts the average localization accuracies at several network densities. The top line indicates error when the bare APS algorithm is simulated against an empirical profile. The two middle lines indicate error when the algorithm is augmented to handle outliers and loss, respectively. The bottom line indicates error when APS is simulated against theoretical ranging data derived from a parametric function.

4.1 Outliers

The normality plot in Figure 5 shows that our ultrasonic ranging data has heavy tails and both positive and negative outliers¹. Figure 6 shows a time series of ranging estimates between two nodes that are one foot apart. The negative outliers are false positives; they represent detections of ultrasound before the ultrasound actually arrived, possibly due to random noise such as keys jingling or a car door slamming. The positive outliers are detections well after the signal has arrived, possibly caused by a low signal to noise ratio.

¹in a normality plot, if the data is Gaussian distributed it will fall in a straight line

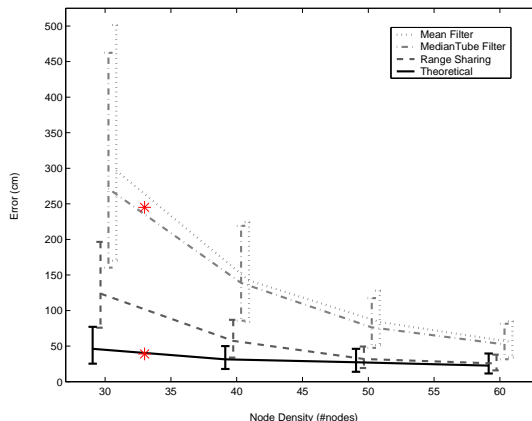


Figure 4: **Effect of Density** median localization error from 100 simulations on a 12m x 12m area at each of four network densities. Error bars indicate upper and lower quartiles of error. The four lines from top to bottom represent simulation on empirical data, empirical data with medianTube filtering, empirical data with range sharing, and theoretical data drawn from an error model of 5m range and 10cm error.

While the mean range estimate is sensitive to outliers, the median is not. One way to eliminate outliers is to take multiple samples and to eliminate all readings too far from the median. Because time of flight has a bias toward having positive errors, we choose the minimum value that falls within a predefined range of the median to be the point estimate of the entire time series. This non-linear filter is thus designed specifically for the outlier distribution observed with our ranging implementation.

We call this filter *medianTube* and have found through testing in simulation that a tube width of about 10 centimeters performs significantly better than a simple median filter. The maximum error is reduced from nearly 2 meters to about 0.4 meters, and the upper quartile of the error is about 10 centimeters. Both the averaged and filtered estimates are shown in Figure 7, and the effect of medianTube on localization is shown as the second line from the top in Figure 4. MedianTube reduces average localization error over all densities by approximately 20cm, beginning to close the gap between the theoretical simulation results and the empirical simulation.

4.2 Loss

In our environment and with the APS algorithm, loss is a much more significant source of localization error than outliers. Figure 8 shows that the probability of receiving a ranging estimate decreases over distance, as measured in four differ-

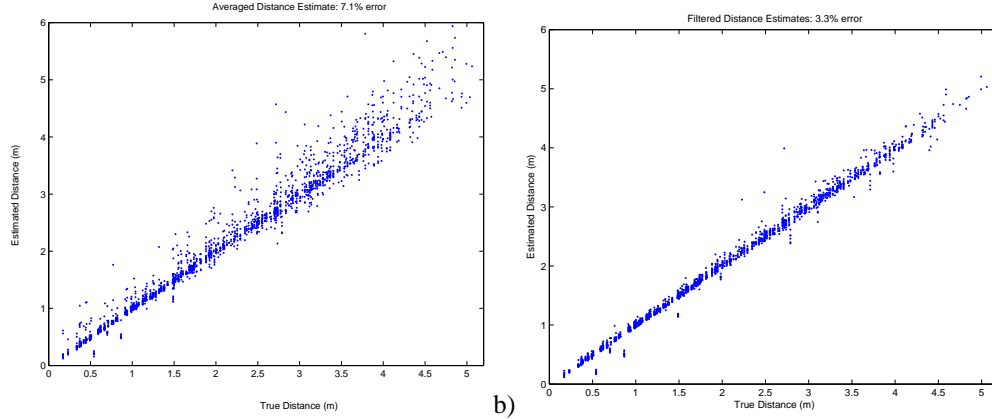


Figure 7: **MedianTube Results** a) Distance estimates after averaging 10 readings. b) Distance estimates after running medianTube on 10 readings.

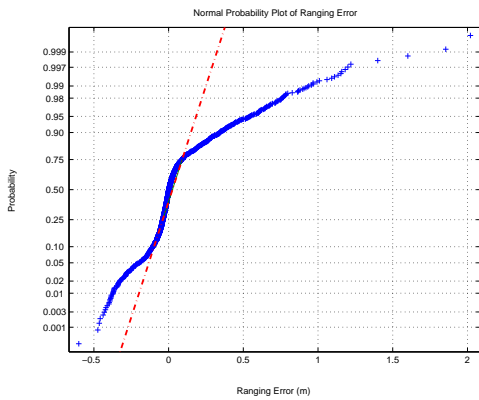


Figure 5: **Normality Plot** If Gaussian data were plotted with this special Y-axis it would form a straight line. The curves in the line of empirical points indicate heavy tails and outliers.

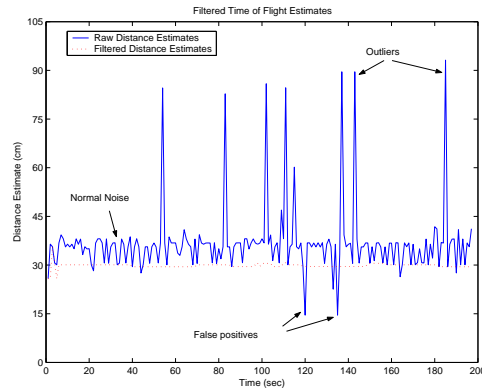


Figure 6: **MedianTube Filter** The medianTube filter chooses the minimum value within a small range of the median. This eliminates outliers and false positives.

ent environments: grass, pavement, carpet, and free space. This loss can reduce network connectivity by a factor of two or more and therefore has a dramatic effect on the APS algorithm. Furthermore, the differences between the curves indicates that APS will perform differently in different environments.

To ameliorate the effects of loss, Calamari introduces a phase of *range sharing* between neighbors. After all ranging has completed, each node transmits its set of distance estimates to all neighbors. If two nodes already have distance estimates to each other, they both use the minimum of the two estimates since the higher one is more likely to be an outlier. If only one of the two nodes has an estimate to the other, both nodes use the same estimate. Range sharing increases ranging connectivity by leveraging the duplicated ranging between all pairs of nodes. Evaluation in simulation shows that this asymmetrical sharing

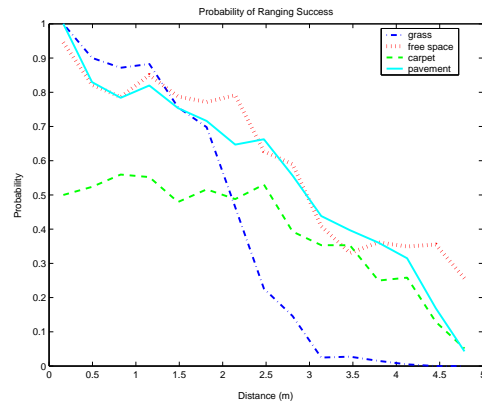


Figure 8: **Loss Rates** This graph shows the loss rates over distance in four different environments. The differences in the curves indicate that our algorithm will behave differently in each environment.

protocol performs better with our particular noise distribution than more straight forward averaging of distance estimates.

Range sharing alleviates the effect of loss with only one or two extra messages per node. The effect of range sharing is more pronounced than the effect of outlier filtering, which speaks to the sensitivity to loss of this particular localization algorithm. As shown in Figure 4, range sharing reduces the error of empirical simulation results to just above that of theoretical model-based localization. More importantly, the *variation* in errors after range sharing is up to three times smaller than results with mere outlier filtering. Thus range sharing is a critical step in achieving predictability.

Instead of using an upper bound to characterize range, we can use what we call *effective range*, defined as

$$\text{effective range} \triangleq \int_{r=0}^{r_{max}} \Pi * r^2 * p(S|r)$$

where $p(S|r)$ is one minus the probability of loss at distance r . This definition is essentially an integral over the probability of obtaining a neighbor at a given distance. This metric is derived from the intuition that it is the total number of neighbors or *node degree* that influences localization, not the distance of the farthest neighbor.

5 Pre-deployment Verification

Deployments are very costly and time consuming. Even for a small deployment, we would like to estimate the probability of success before deployment actually takes place. This is increasingly important for extremely large networks or mission critical situations that can only be deployed once.

Sensor networks are often deployed in pseudo-random patterns such as a grid formation with Gaussian noise on each grid location or uniformly at random over a square region. Because random distributions can result in very diverse topologies, some of the resulting networks will localize well while others will not. We can predict the probability that a deployment will succeed by generating hundreds of topologies from the same random parameters. On each topology, we then simulate the localization algorithm against the empirical profile of the deployment environment. If, for example, 60% of the topologies localize to the desired level of accuracy, we say that the deployment has a 60% chance of success.

This simulation technique can also be used to choose deployment parameters to maximize localization accuracy and the likelihood of successful localization. For example, in Section 7, we will need to localize a network over a 13x13m area to an accuracy of about 0.5m. These specifications are suitable for applications such as the

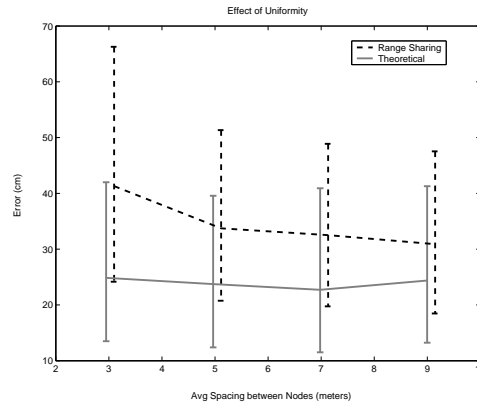


Figure 9: **Anchor Density Effect Simulations** on empirical data reveal that, for networks of 50 nodes in a 12m x 12m area, more than 4 anchors nodes do not significantly improve results.

Pursuer-Evader Game, in which the sensor network assists an autonomous robot in pursuit of another robot who's position is being tracked by the network [17]. Before deployment, we need to determine how many nodes to deploy and how many anchor nodes are required.

Figure 4.a summarizes the results of our system simulated on 30, 40, 50, and 60 node networks against an empirical profile of ranging on a paved surface. Each point represents the median error of 100 simulations and the error bars indicate upper and lower quartiles of error. Results indicate that we need 50 nodes to yield approximately a 75% chance of achieving 0.5m median error. Similarly, Figure 4.b summarizes the results of simulations with 3, 5, 7, and 9 anchor nodes. Results indicate that localization accuracy does not improve with more than 4 anchor nodes. These results determined the deployment parameters used in the deployment presented in Section 7. Similar simulations were run to determine the best environment and deployment pattern for the deployment.

It is important to note that simulations against ranging data generated from theoretical models yielded optimistic predictions of system performance. For example, from Figure 4.a we see that theoretical data would suggest that we deploy our system with only 30 nodes to achieve 0.5m accuracy. Actually doing so would have resulted in 1.25m error, 150% more than expected.

6 Unexpected Problems

While our simulation environment captures real-world ranging characteristics, it does not capture all aspects of a deployment. In this section we describe two problems we encountered during the deployment that were not predicted by our simulation.

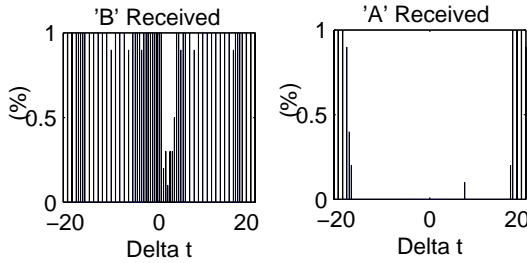


Figure 10: **Shadowing** Nodes *A* and *B* both transmit with a time difference of Δt . When $|\Delta t| < 17$, the messages overlap but a third node can still hear *B*'s message with almost 100% reliability.

6.1 Ranging Collisions

Raw ultrasound pulses are not differentiable and collisions between them cannot be detected. Several techniques have been proposed to avoid ranging errors during collisions. For example, each node could encode a signature in the ultrasound pulse with frequency or amplitude modulation, allowing the detection of ultrasound collisions and even allowing the reception of multiple ultrasound signals simultaneously, as seen in [3]. Another solution is to envelope the ultrasound pulse in a radio message such that a collision between ultrasound signals can be detected through the collision in radio messages. This technique, first proposed in [11] and used in many ultrasound ranging implementations since, relies on the fact that no messages are received during a radio collision, in which case all ultrasound information is ignored.

Calamari initially employed the second of these techniques because of its simplicity. However, collisions were still a problem because with the FSK radio used on the mica2dot, radio collisions are *not* always detectable; when two nodes transmit simultaneously, a third node will often receive one message but not the other, even if both are received with almost the same signal strength. In other words, with FSK radios, RF collisions do not always result in corrupted packets. This phenomenon is known as *capture* as described in [21] and is possible though very uncommon in ASK radios such as those used in [11].

To test the extent of this phenomenon, we performed an experiment in which three nodes *A*, *B*, and *C* were approximately one meter from each other. We monitored the reception of messages at *C* while both nodes *A* and *B* would transmit with a time difference of Δt . Δt was varied from -20ms to 20ms. As shown in Figure 10, the messages from both nodes are received when $|\Delta t| > 17$ ms, the length of the packet. When

$|\Delta t| < 17$ ms the packets collide and neither packet should be expected to be received. However, *B*'s messages are still received at *C* with almost 100% reliability. This means that *B*'s messages are being received during a collision with *A*'s messages.

Capture presents a serious problem for detecting ultrasound collisions. We implemented an application-level MAC protocol in which each node sent ranging messages in batches of twenty with a small random delay between each message. Thus, the probability that every message in a batch from one node collides with every message in a batch from another node decreases exponentially as the length of the batch grows. This allows receiving nodes to discard data from ranging collisions: any node that hears messages from two overlapping batches can discard all ranging messages from both batches. Furthermore, if a node hears another node sending a ranging message, it backs off for the entire duration of the batch, not just the ranging message. With this technique, all messages toward the end of each batch can usually be sent without any collisions.

6.2 Correlated Errors

One limitation of the simulation techniques described in Section 3 is that it does not capture the possibility of correlated errors. For example, tree in the center of the network or a dead or error prone node would cause ranging errors that are all correlated in space. A period of strong wind or rain would cause ranging errors that are correlated in time. Correlated errors are not captured in our data collection process and, even if they were, the correlations are not accounted for by our statistical sampling process used in simulation. Because we are deploying in an open space at a known time, we expected this limitation to be inconsequential.

Our final deployment of 49 nodes, however, was large enough that a 5% hardware failure rate became significant. Of the 49 nodes in the deployment, only 46 of them were fully functional. Some nodes would always estimate a distance of zero to all other nodes, even those on the other side of the network. A single such node would wreck havoc in the shortest path algorithm and all such nodes were turned off. Other nodes did not receive any ranging estimates from any other nodes, leaving holes in the connectivity graph and increasing all shortest path estimates. Incorporating dead and erroneous nodes into the simulation might have suggested using higher density networks to compensate for dead nodes, and any node with suspicious ranging estimates could have been designed to simply cut itself out of the network. This is currently reserved for future work.

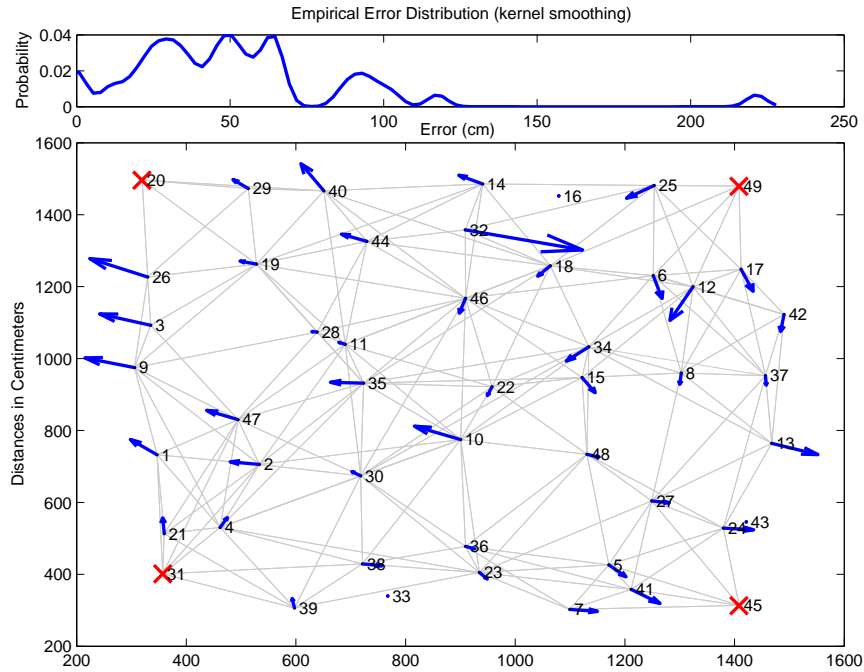


Figure 12: **A Single Deployment** The top graph is a kernel smoothing of the error distribution and the bottom graph shows the actual node positions with black dots and their location errors as arrows. The anchor nodes are indicated by "X"'s and the gray lines indicate ranging connectivity. Nodes 33, 16, and 43 were dead nodes. The median error for this run was 47.8cm.

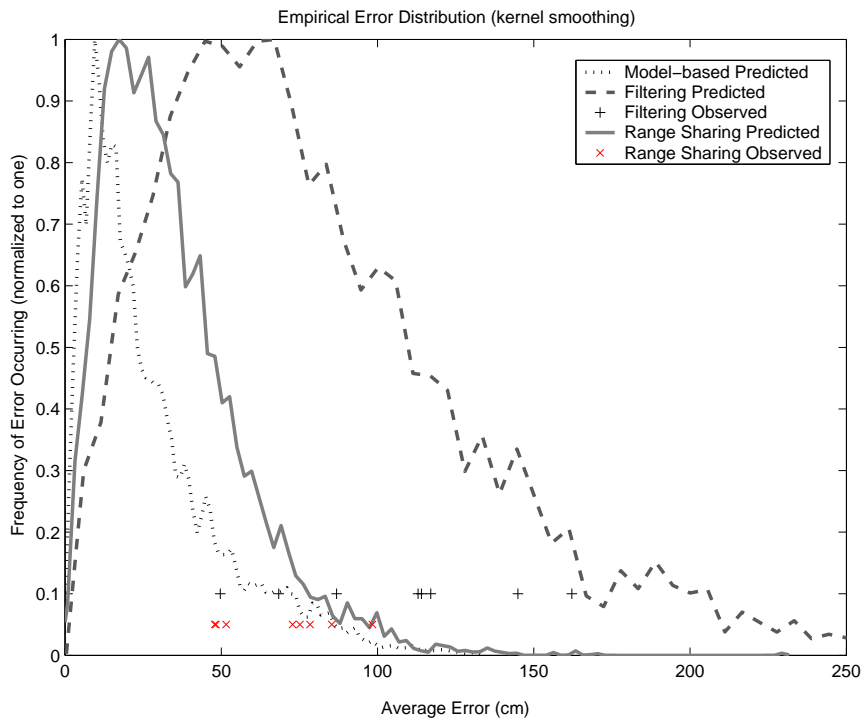


Figure 13: **Predictions and Observations** The '+'s and 'x's indicate observed average error before and after range sharing, respectively. The three curves represent the errors distributions seen in simulation using 1) empirical data with medianTube 2) empirical data with range sharing and 3) theoretical ranging data generated from a model with 5m maximum range and 10cm noise.

	Lower Quartile (cm)	Median (cm)	Upper Quartile (cm)
model-based prediction	11.0	20.6	37.7
medianTube prediction	43.1	69.7	107.3
medianTube observed	42.6	85.2	140.8
range sharing prediction	17.8	29.6	45.4
range sharing observed	32.5	53.4	86.8

Table 1: **Localization Errors** This table presents the localization error quartiles predicted by theoretical ranging data with 5m maximum range and by empirical data with and without medianTube and range sharing. The error quartiles observed during the 8 deployments, both with and without range sharing, are shown in bold.

	Lower Quartile (cm)	Median (cm)	Upper Quartile (cm)
model-based prediction	7.5	17.7	32.1
medianTube prediction	11.9	38.9	82.2
medianTube observed	18.5	45.5	106.0
range sharing prediction	5.1	12.8	28.8
range sharing observed	11.0	30.5	67.5

Table 2: **Shortest Path Errors** This table presents the shortest path error quartiles predicted by theoretical ranging data with 5m maximum range and by empirical data with and without medianTube and range sharing. The error quartiles observed during the 8 deployments, both with and without range sharing, are shown in bold.



Figure 11: **Final Deployment** The final deployment involved 49 nodes over a 13x13m area on a paved surface.

7 Experimental Results

We performed a real deployment to evaluate how well the empirical simulation techniques could predict the actual outcome of a real deployment, and to test whether the deployment methodologies that we were using actually help produce localization results that meet our application requirements.

To avoid doing this single deployment test on an especially good or bad topology, we generated 100 random topologies and chose the one that yielded median average error in simulation. These topologies were generated for network with 49 nodes and 4 anchors in a 13x13m grid with random noise added to the grid. The hop

count of the longest shortest path in the selected topology network was 8, although longer paths appeared in the real deployment.

The main deployment took place outdoors in the parking lot shown in Figure 11 under weather conditions similar to those in which the original ranging data was collected. After we measured the topology and placed the nodes, we executed the localization system on the network eight times, each time calculating shortest paths and positions both with and without range sharing. The sixteen final results are shown in Figure 13 against a backdrop of prediction made by simulations with several different types of ranging data. The results of a single run after range sharing are shown in Figure 12 for closer analysis.

The median of the observed deployment errors before and after range sharing are 85cm and 53cm, respectively. Using this same topology, 100 simulations predicted errors of 70cm and 30cm. While a strong correlation is evident, the predicted statistics do not match the observed behavior exactly. While this could be an artifact of the small sample size, it could also be an indication that there are still factors that are unaccounted for in the simulations.

A deeper analysis reveals that the loss rates in the deployment are slightly higher than those in the simulations. Before the range sharing, the median degree (number of neighbors) of each node in simulation and in the real deployment were 7 and 5, respectively. Over all, the simulated topologies had an average of 362 edges in the ranging connectivity graph while the real de-

ployments only had 271. After the range sharing, the node degrees were bumped up to 10 and 7, respectively. Consistency checking added an average total of 139 new edges to the simulated connectivity graphs but only 78 to those in the real deployment.

The discrepancy in connectivity seems to have been the cause of the difference between simulation predictions and empirical observation. Indeed, the median length of the shortest paths in the true deployments was one hop longer than that of the simulations, causing a sharp increase from the predicted 13cm median shortest path error to an observed 31cm. This is likely to be at least partially a result of the fact that three of the nodes in the final deployment were dead. Nonetheless, a discrepancy of only 20cm between predicted and observed localization is much smaller than we would have seen with traditional simulation techniques.

8 Discussion

In this paper we present a deployment process used for an ultrasound-based localization system. We take an existing localization algorithm and carry it through the steps of hardware implementation, distributed software implementation and final deployment. The framework for our deployment methodology is a combination of new data collection and simulation techniques that make it possible to simulate a localization system in the context of real-world noise. This enables the design and evaluation of a system with respect to the noise of a specific environment. Furthermore, these simulations can be used to choose deployment parameters and estimate the probability of successful localization before the deployment actually takes place. This is especially important for very large or mission critical deployments in cases where field experimentation is not possible.

While we evaluate these simulation techniques in the context of the deployment of an ultrasound-based localization system, the scope can easily be extended to other contexts. The techniques extend in a straightforward way to other ranging modalities and would also extend to algorithm evaluation in non-deployment contexts. Most localization algorithms today have only been tested in traditional simulation environments and it is difficult to know how well the simulations predict the behavior of the algorithm in the face of real-world noise. The techniques presented in this paper can be used to evaluate and compare the many localization algorithms in the literature. Similarly, most networking simulations today use theoretical radio models to model radio collisions and connectivity. These simulations could benefit from empirical profiling of RF communication channels in outdoor environments.

Acknowledgements

This work is funded in part by the National Defense Science and Engineering Graduate Fellowship, the UC Berkeley Graduate Opportunity Fellowship, the DARPA NEST contract F33615-01-C-1895, and Intel Research. Special thanks to Joe Polastre and Rob Szwedczyk for help with hardware design and to Cory Sharp and the PEG crew for help with software and deployment.

References

- [1] Calamari web page. <http://www.cs.berkeley.edu/~kamin/calamari/index.html>.
- [2] P. Bahl and V. N. Padmanabhan. RADAR: An in-building RF-based user location and tracking system. In *INFOCOM (2)*, pages 775–784, March 2000.
- [3] L. Girod and D. Estrin. Robust range estimation using acoustic and multimodal sensing. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2001.
- [4] A. Harter, A. Hopper, P. Steggle, A. Ward, and P. Webster. The anatomy of a context-aware application. In *Mobile Computing and Networking*, pages 59–68, 1999.
- [5] J. Hill and D. E. Culler. Mica: a wireless platform for deeply embedded networks. *IEEE Micro*, 22(6):12–24, nov/dec 2002.
- [6] B. Hofmann-Wellenho, H. Lichtenegger, and J. Collins. *Global Positioning System: Theory and Practice*. Springer Verlag, fourth edition, 1997.
- [7] K. Langendoen and N. Rajjers. Distributed localization in wireless sensor networks: a quantitative comparison. *Computer Networks*, 43(4):499–518, November 2003.
- [8] L. Navarro-Serment, C. Paredis, and P. Khosla. A beacon system for the localization of distributed robotic teams. In *The International Conference on Field and Service Robotics*, Pittsburgh, PA, August 1999.
- [9] D. Niculescu and B. Nath. Ad Hoc Positioning System (APS). In *IEEE GLOBECOM*, pages 2926–2931, 2001.
- [10] N. Patwari, A. Hero, M. Perkins, N. Correal, and R. O’Dey. Relative location estimation in wireless sensor networks. *IEEE Transactions on Signal Processing, Special Issue on Signal Processing in Networks*, 51(8):2137–2148, August 2003.
- [11] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan. The cricket location-support system. In *Sixth Annual ACM International Conference On Mobile Computing and Networking (MOBI-COM)*, 2000.
- [12] C. Savarese. Robust positioning algorithms for distributed ad-hoc wireless sensor networks. Master’s thesis, University of California at Berkeley, 2002.
- [13] A. Savvides, C.-C. Han, and M. B. Strivastava. Dynamic fine-grained localization in ad-hoc networks of sensors. In *Mobile Computing and Networking (MobiCom)*, pages 166–179, 2001.

- [14] A. Savvides, H. Park, and M. Srivastava. The bits and flops of the n-hop multilateration primitive for node localization problems. In *First ACM International Workshop on Sensor Networks and Applications*, 2002.
- [15] A. Savvides, H. Park, and M. B. Srivastava. The bits and flops of the n-hop multilateration primitive for node localization problems. In *First ACM International Workshop on Sensor Networks and Applications*, September 2002.
- [16] Y. Shang, W. Ruml, Y. Zhang, and M. P. J. Fromherz. Localization from mere connectivity. In *MobiHoc*, June 2003.
- [17] C. Sharp, C. Karlof, N. Sastry, A. Woo, S. Schaffert, and S. Sastry. Design and Implementation of a Sensor Network System for Vehicle Tracking and Autonomous Interception. European Workshop on Wireless Sensor Networks (EWSN), 2005.
- [18] M. L. Sichitiu, V. Ramadurai, and P. Peddabachari. Simple algorithm for outdoor localization of wireless sensor networks with inaccurate range measurements. In *International Conference on Wireless Networks 2003*, pages 300–305, 2003.
- [19] S. Simic. A distributed algorithm for localization in random wireless networks. submitted to *Discrete Applied Mathematics*, 2002.
- [20] R. Stoleru and J. A. Stankovic. Probability grid: A location estimation scheme for wireless sensor networks. In *SECON*, 2004.
- [21] A. Woo, K. Whitehouse, F. Jiang, J. Polastre, and D. Culler. Capture: Collision detection and recovery. Technical Report UCB//CSD-04-1313, UC Berkeley, March 2004.