Self-Configuring Localization Systems: Design and Experimental Evaluation

NIRUPAMA BULUSU†
University of California at Los Angeles
JOHN HEIDEMANN
University of Southern California/Information Sciences Institute
DEBORAH ESTRIN and TOMMY TRAN
University of California at Los Angeles

Embedded networked sensors promise to revolutionize the way we interact with our physical environment and require scalable, ad hoc deployable and energy-efficient node localization/positioning.

This paper describes the motivation, design, implementation and experimental evaluation (on sharply resource-constrained devices) of a *self-configuring* localization system using radio beacons. We identify beacon *density* as an important parameter in determining localization quality, which saturates at a transition density. We develop algorithms to improve localization quality by (i) automating placement of new beacons at low densities (HEAP) and (ii) rotating functionality among redundant beacons while increasing system lifetime at high densities (STROBE).

Categories and Subject Descriptors: C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*wireless communication*; C.3 [**Special-purpose and Application-Based Systems**]: Real-time and embedded systems; C.4 [**Performance of Systems**]: Design Studies

General Terms: Algorithms, Design, Experimentation, Measurement, Performance, Reliability

Additional Key Words and Phrases: location, localization, self-configuration, sensor networks

1. INTRODUCTION

Recent technological advances have fostered the emergence of small, low-power devices that integrate micro-sensing and actuation with on-board processing and wireless communications capabilities. When deployed in large numbers and embedded deeply within large-scale physical systems, these devices gain the ability to measure aspects of the physical environment in unprecedented detail. Through distributed coordination, pervasive networks of micro-sensors and actuators promise to revolutionize the way we understand and construct complex physical systems [Estrin et al. 1999].

† Corresponding Author
N. Bulusu, D. Estrin and T. Tran can be reached at: University of California at Los Angeles, Laboratory for Embedded Collaborative Systems, Room 3440 Boelter Hall, Los Angeles, CA 90095. phone: (310) 206 3925. fax: (508) 437 8815. email: {bulusu;estrin;ttran}@lecs.cs.ucla.edu
J. Heidemann can be reached at: USC/Information Sciences Institute, Suite 1001,4676 Admiralty Way, Marina del Rey, CA 90292-6695. phone: (310) 448 8708. fax: (310) 823-6714. email: johnh@isi.edu
Permission to make digital/hard copy of all or part of this material without fee for personal or classroom use provided that the copies are not made or distributed for profit or commercial advantage, the ACM copyright/server notice, the title of the publication, and its date appear, and notice is given that copying is by permission of the ACM, Inc. To copy otherwise, to republish, to post on servers, or to redistribute to lists requires prior specific permission and/or a fee.
© 20TBD ACM 1529-3785/20TBD/0700-100001 $5.00

There are many potential applications of sensor networks: physiological monitoring; environmental monitoring (air, water, soil, chemistry); condition based maintenance; smart spaces; military surveillance; precision agriculture; transportation; factory instrumentation and inventory tracking. In this paper, we focus on densely distributed, physically coupled and wireless sensor networks.

Three characteristics distinguish wireless networked embedded computing systems (or wireless sensor networks) from traditional desktop based computing systems. First, individual devices will be much smaller and cheaper and untethered. Consequently, they will have modest processing and communications capabilities, and much limited energy on board. Second, they will be more numerous. Finally, the ratio of devices per human will be several orders of magnitude higher than in desktop computing. Such systems must be able to operate unattended with minimal configuration and re-configuration requirements. Such considerations mandate that these systems on the whole must be scalable, ad hoc deployable and energy-efficient.

Localization may be defined as the problem of estimating the spatial relationships among objects. Localization is fundamental to coordination amongst various embedded networked sensors as these systems are tightly coupled to the physical world. Because sensor data are intrinsically associated with the physical context of the phenomena being sensed, *spatial coordinates* are often a natural way to name data. Spatial coordinates are also employed by collaborative signal processing algorithms (e.g. beamforming) that combine data from multiple sensor nodes for such tasks as target tracking. Furthermore, geographic assistance in ad hoc routing promises significant reductions in energy consumption [Karp and Kung 2000; Xu et al. 2001]. However, existing localization systems such as GPS[Hofmann-Wellenhoff et al. 1997] may be used in some nodes, but cannot always meet the operational (low power), environmental (indoors) and cost constraints of all the deployed nodes.

Localization has been studied for many years as a classical problem in many disciplines (and under many names!), including the robot localization problem in mobile robotics [Thrun et al. 2001], the motion tracking problem in virtual reality systems [Welch et al. 1999], in navigation systems (VOR [VOR] and the Global Positioning System (GPS)[Hofmann-Wellenhoff et al. 1997]) and identifying user location in cellular networks (RadioCamera [US Wireless Corporation]).

When engineering localization systems for these applications, environmental dependence has proven to be a major challenge. The nature of the environment (such as indoors or outdoors, temperature, pressure, weather, objects and various sources of interference) influences not only the characteristics of the sensors used for localization but also the magnitude and type of measurement errors. Traditionally, this has been addressed through extensive environment-specific calibration and configuration of the centrally controlled, tightly coupled localization system (as in HiBall [Welch et al. 1999], Radar [Bahl and Padmanabhan 2000] and RadioCamera [US Wireless Corporation]) and the use of sophisticated, memory and compute-intensive probabilistic position-estimation algorithms (such as Monte Carlo Localization [Thrun et al. 2001]).

These approaches are however not suited for large scale sensor networks. Sensor networks require node localization, but as mentioned previously, under far severe node-level resource constraints (limited energy, bandwidth, memory and processing) and system level operational constraints. Localization approaches that can reconcile the needs of large ad hoc, sensor networks are wireless and distributed such as [Bulusu et al. 2000], Cricket

[Priyantha et al. 2000], AHLoS [Savvides et al. 2001], [Girod 2000] and SpotON [Hightower et al. 2000] and lack centralized coordination and control. Our thesis is that such a localization system must also *self-configure*, *i.e.,* autonomously measure and adapt its properties to environmental conditions (rather than rely on design-time pre-configuration or manual reconfiguration) in order to achieve ad hoc deployment and robust, unattended operation in any environment.

In sensor networks, node localization can leverage having a few nodes at known positions (also known as *beacons*) and compute the positions of other nodes relative to the coordinate system defined by the beacons; or nodes can form a completely independent coordinate system. For reasons we describe in Section 3 we chose an approach based on beacons. In this paper, we describe the motivation, design, implementation and evaluation of a self-configuring localization system based on beacons. Our key contributions are as follows.

—To address beacon deployment issues, we introduce the novel concept of *self-configuring beacon networks*.

—We identify *density* as an important parameter in characterizing localization quality, develop a methodology and propose two algorithms for system self-configuration based on beacon density. For sparse and medium density deployments, we propose the HEAP algorithm to detect regions with poor localization, and select candidate points for placing new beacons. For dense beacon deployments, we propose the STROBE algorithm. STROBE enables densely deployed beacons to coordinate without self-interference and opportunistically conserve energy.

—We also evaluate and demonstrate the effectiveness of our solutions. We use simulations to explore in detail the implications of several design choices. We present the *measured performance* of an implementation of a radio based localization system that demonstrates that the granularity of localization improves by adding beacons at points selected by the HEAP algorithm. We also verify and validate the simulated performance of the STROBE algorithm using experimental emulations.

2. BACKGROUND

Localization is by nature an interdisciplinary problem involving several areas of computer science and relevant to many kinds of engineering systems. Consequently, research has proceeded on both the systems and algorithmic fronts in computer science. In this section, we review why state-of-the-art developments in these areas do not meet the requirements and motivate our approach.

2.1 Systems

The design of a localization system is largely influenced by application requirements - such as the requirement highly accurate or real time position estimation. The system can be either *tightly coupled* (beacons that are wired to a centralized controller and placed at fixed positions) or *loosely coupled* (beacons that are wireless and coordinate in a completely decentralized manner with no central control).

2.1.1 *Tightly coupled systems.* There are many tightly coupled systems such as the ActiveBat [Ward et al. 1997] developed for sentient computing applications and the HiBall

Table I. Notation used in this paper to describe localization and beacon placement algorithms.

SYMBOL	NAME	DEFINITION
B	Beacon	A device that knows its position. (and is presumed to be static.)
C	Client	A device whose position is unknown. (can be static or mobile.)
$\mathcal{BP}$	Beacon Placement	An assigned placement of beacons.
N		Total number of deployed beacons.
A	Area	Total area in which beacons are deployed.
$position(B)$		Position of a beacon B.
$Range$	Range	Nominal transmission range of the beacons.
ρ	Beacon Deployment Density	Number of beacons per unit area.
μ	Beacons Per Neighborhood *or* Beacons Per Nominal Radio Coverage Area (bpnrca)	Number of beacons present in a nominal radio transmission coverage area of $\pi \cdot Range^2$.

$$\rho = \frac{N}{A} \tag{1}$$

$$\mu = \rho \cdot \pi \cdot Range^2 \tag{2}$$

Tracker [Welch et al. 1999] designed for virtual reality applications. These applications have high accuracy and real-time tracking requirements.

Problems of time synchronization and coordination amongst beacons are easily resolved because these systems are wired and have a centralized controller. These systems therefore achieve high accuracy. But the drawback is that the centralized position estimation limits the number of devices these systems can simultaneously track (HiBall). Secondly, wiring significantly impedes deployment. A key research challenge in these systems is achieving similar granularity outdoors where deployment cannot be controlled and wiring may be infeasible.

2.1.2 *Loosely coupled systems.* Motivated by deployment concerns, recently proposed localization systems [Bulusu et al. 2000], Cricket [Priyantha et al. 2000] and AHLoS [Savvides et al. 2001] are decentralized and completely wireless. They sacrifice the accuracy of tightly coupled systems for ease of deployment, and scalability to large numbers of devices. They rely on a system of beacons, each of which periodically transmits an advertisement containing its position. Clients compute their position based on the advertisements they receive.

Because beacons are wireless and deployed in an ad hoc manner, beacon coverage is not guaranteed. Due to the lack of centralized control, there is no explicit coordination amongst beacons. Thus beacons can contend and self-interfere when emitting a signal (radio, acoustic etc.). These problems need to be addressed.

2.2 Algorithms

Algorithmic developments in localization focus on two complementary problems - robust position estimation, and optimal node placement.

2.2.1 *Position estimation.* In the field of mobile robotics, localization has been referred to as "the most fundamental problem to providing a mobile robot with autonomous

capabilities" [Cox 1991]. Environmental obstructions such as walls, moving people and objects, can greatly interfere with the sensing capabilities of a mobile robot. Consequently, the focus here lies on robust position estimation in real-time through probabilistic localization techniques that account for unpredictable sensing error.

An interesting development in probabilistic localization algorithms for mobile robot navigation has been Monte Carlo Localization (MCL) [Thrun et al. 2001]. MCL algorithms represent a robot's belief by a set of weighted hypotheses (samples), which approximate the posterior under a common Bayesian formulation of the localization problem. These algorithms are computationally efficient, versatile, resource-adaptive and robust under a range of circumstances. However, these algorithms have been designed to localize a single mobile robot (with PC-class computational hardware) with respect to its environment. Consequently, they have not addressed issues of scalability or hardware constraints.

Doherty has proposed convex optimization techniques [Doherty et al. 2001] for solving the position estimation problem in sensor networks in an off-line, centralized manner. The advantage of this approach is that it requires very few references (or beacons) since all system constraints are solved globally. However, this algorithm is not very robust to failures - when there are ambiguities in measurements.

Savvides *et al* have explored iterative techniques for robust position estimation in sensor networks [Savvides et al. 2000] . Iterative techniques incur additional energy costs in communication, and are not guaranteed to be completely fault tolerant.

2.2.2 *Optimal Placement.* Optimal placement problems have been studied in various contexts by researchers including facility location [Charikar et al. 1999] and pursuit evasion problems in robotics [Guibas et al. 2000].

In robotics, the classical Art gallery problem formulation has been used by researchers to address questions of placement. In the "art-gallery" analogy, the robot's goal is to move from one position to another to maximize visual coverage of its surroundings, as a human might try to do in a gallery. A complementary set of approaches addresses the pursuit-evasion problem in which a robot tries to move so as to evade observation or capture by mobile trackers. However these approaches are based on modeling the environment as a polygon and are best suited for vision-based localization systems. They account for neither the noise nor the wide variety of terrain conditions one would expect to encounter for ad hoc sensor networks.

Facility Location [Charikar et al. 1999] problems are a well known class of theoretical computer science problems and have been the subject of extensive research over the past thirty-five years. In these facility location problems, there is a set of locations, where the cost of building a facility at location i is $f(i)$; furthermore, there is a set of client locations (such as stores) that require to be serviced by a facility, and if a client at location j is assigned to a facility at location i, a cost of $c(i, j)$ is incurred. The objective is to determine a set of locations at which to open facilities, so as to minimize the total facility and assignment costs. Since these problems are NP-hard, it is unlikely that there exist efficient algorithms to find optimal solutions. Instead, the focus has been on designing algorithms that are guaranteed to find solutions within a particular factor of the optimum. Solutions are based on linear relaxations to the natural integer programming formulations that yield extremely good lower bounds. Unfortunately, these algorithms do not provide proper cost models to represent beacon connectivity.

Researchers have recognized that these systems will be deployed at large in an ad

hoc fashion, without controlling the placement of each and every node. Instead, they have focused on developing techniques to identify problems in a deployed sensor field. Meguerdichian *et al* [Meguerdichian et al. 2001] have also proposed algorithms for coverage in wireless ad hoc sensor networks given global knowledge of node positions using Voronoi diagrams [Aurenhammer 1991] to compute maximal breach paths and find gaps.

The techniques developed for system deployment through optimal node placement are a) not scalable to very large sensor networks, b) not suitable for rapid deployment and c) not generalizable to a variety of environments and systems, suffering unknown and unpredictable radio propagation vagaries. It is virtually impossible to adapt to such terrain and propagation uncertainties and compute a satisfying beacon placement to achieve uniform localization granularity across the terrain using a purely algorithmic approach.

2.3 Summary

Despite these state of the art developments in systems and algorithms, a fundamental challenge remains. How can we achieve robust localization under the following conditions?

—Scalability is paramount and motivates localization systems that are loosely coupled on the global scale such as [Bulusu et al. 2000] and [Priyantha et al. 2000].

—Computational and hardware resources are modest and preclude more sophisticated position estimation algorithms such as [Thrun et al. 2001].

—System and environment conditions vary greatly over time and space and preclude design-time pre-configuration algorithms such as [Meguerdichian et al. 2001].

As we discussed in the introduction, one key distinguishing characteristic of sensor networks is the extremely high ratio of devices per human and the consequent need for robust, unattended operation. Consequently, making localization self-configuring in response to environmental and system conditions becomes very important. In the next Section, we develop our methodology for making localization self-configuring.

3. SELF-CONFIGURING BEACON NETWORKS

In this Section, we describe our intuition for self-configuring beacon networks and develop our beacon density-based methodology for achieving it.

3.1 Using Beacons for Localization

Our localization system described in Appendix A uses beacons. Besides our approach, several proposed localization systems rely on beacons [Priyantha et al. 2000], [Ward et al. 1997], [Welch et al. 1999]. Localization systems using some beacons have two advantages over localization systems with no beacons. First, having beacons spatially distributed throughout the geographical region lets devices compute their location in a scalable, decentralized manner. Second, even when the application permits off-line, centralized position-estimation algorithms (as in [Doherty et al. 2001]), both the convergence and estimation accuracy can be significantly improved by having some nodes as beacons [Doherty et al. 2001].

These beacons constitute the underlying infrastructure of the localization system. There are two major configuration and deployment concerns when beacons are used.

—*Beacon Configuration.* Each beacon needs to be configured with its spatial coordinates during deployment. Automating this process is important for large scale and highly dense beacon deployment. In an outdoor setting, we assume that beacons can infer their

position through GPS. In an indoor setting, we believe that initial beacon placement will be structured. Only a few beacons will need to have their positions assigned manually, the rest can exploit this structure (for example, in a rectangular grid) in beacon placement to infer their coordinates. This is the approach used in fact in the HiBall tracker [Welch et al. 1999].

—*Beacon Placement.* How many beacons do we need? Where should they be placed? The beacon density and placement are important in influencing the overall localization quality. Uniformly dense placement is good and has its benefits, it is not adequate.

3.2 Impact of Beacon Density

We start by considering the impact of beacon density on the quality of localization in these systems.

3.2.1 *Characterizing Beacon Density.* A classical notion of node density is the deployment density.

Beacon deployment density ρ. denotes the number of beacons per unit area.

$$\rho \ = \ \frac{N}{A} \tag{3}$$

However, this definition does not abstract away the effect of the nominal communication (radio transmission) radius on the perceived density. We have come up with a new density metric that encapsulates the effect of the radio transmission range.

Beacons per neighborhood μ. (also referred to as beacons per nominal radio coverage area $bpnrca$) denotes the number of beacons that exist in a nominal radio transmission coverage area ($\pi \cdot Range^2$).

$$\mu \ = \ \rho \cdot \pi \cdot Range^2 \tag{4}$$

3.2.2 *Impact of Density on Localization Granularity.* For a given beacon placement $\mathcal{BP}$ in a square terrain of area $A = Side \times Side$ and $0 < step \ll Side$. Let us the define the point $P(k,l)$ as follows:

$$P(k,l) \ = \ (k \cdot step, l \cdot step) \quad \forall\, 0 \leq k,l \leq \frac{Side}{step} \tag{5}$$

The quality of localization in the terrain can be characterized in terms of statistical metrics such as the mean and median localization error over various points in the terrain, defined as follows.

$$\textit{MeanErr}(\mathcal{BP}) \ = \ \frac{\Sigma_{k=0}^{\frac{Side}{step}} \Sigma_{l=0}^{\frac{Side}{step}} LE_{\mathcal{BP}}(P(k,l))}{(\frac{Side}{step}+1)^2} \tag{6}$$

$$\textit{MedianErr}(\mathcal{BP}) \ = \ median(LE_{\mathcal{BP}}(P(k,l))) \qquad \forall\, 0 \leq k,l \leq \frac{Side}{step} \tag{7}$$

Figure 1 plots the mean localization error as a function of beacon density. Regardless of actual beacon placement, the localization granularity saturates at a certain threshold beacon density μ_{thresh} (6 in this case). [1]

[1]The saturation density for localization in 2 dimensions can be expected to differ from localization in 3 dimensions.

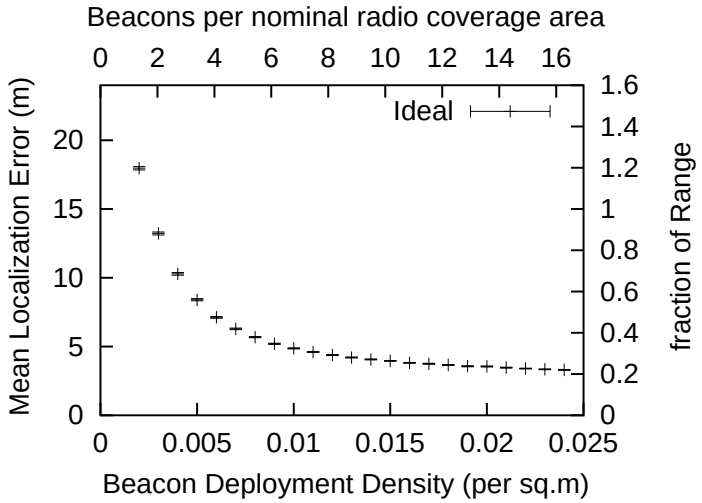


Fig. 1. Mean localization error vs. Beacons per nominal radio coverage area. Localization granularity saturates at a certain number of beacons per neighborhood, around 6 in our case. (Graph based on simulations of 1000 random topologies per beacon density.)

3.2.3 *Impact of Density on Channel Contention and Self-Interference.* Consider a CSMA-like underlying media access protocol for sensor networks. One example of such a media access protocol is SMAC [Ye et al. 2002].

Let us assume the beacons per nominal radio coverage area is μ. Assume that any given instant, the probability of a beacon transmitting an advertisement packet is p. If T_X is the transmission time of an advertisement packet and each beaconing interval is T then

$$p = \frac{T_X}{T} \tag{8}$$

Let $p_{success}$ denote the probability that the packet is successfully received without any interference. Let $p_{collision}$ denote the probability of collision in the wireless system. We can model the channel contention as follows.

Let X indicate the number of beacons that will try to transmit a packet.

$$p_{success} = Pr(X = 1) \tag{9}$$
$$= p \cdot (1-p)^{\mu} \tag{10}$$
$$p_{collision} = 1 - p_{success} \tag{11}$$

This shows us that the probability of packet collision increases exponentially with the beacon density μ. Thus, we cannot simultaneously increase beacon density and maintain the same system responsiveness for localization. [2]

[2] In order to maintain the same collision probability $p_{collision}$ at a higher beacon density, we need to significantly reduce the packet transmission probability p. Since the transmission time of a beacon advertisement packet T_X is fixed, this means that we must correspondingly increase the beaconing interval T. Since the sampling time of a client for its location computation (defined in Appendix A) is directly proportional to T, this means that there is a corresponding increase in location computation latency.

3.2.4 *Two Assertions about Beacon Density.* Thus, we can make two assertions about beacon density in the context of proximity-based localization systems with localized location computation (Chapter A).

(1) Regardless of actual beacon placement, the localization granularity saturates at a certain threshold beacon density μ_{thresh}.
(2) As the beacon density increases, the probability of collisions among competing beacons vying for the same transmission slot increases.

At low and medium beacon densities, the quality of localization suffers due to poor placement of beacons due to various environment and calibration-dependent vagaries in radio signal propagation. For example, different radio transmitters operating at the same power level may have different effective radio transmission ranges if they are not calibrated. Moreover, the same transmitter may experience varying transmission ranges depending on its environment (indoors vs. outdoors). A radio transmitter may have an anisotropic radio coverage pattern, determined by its environment and the obstacles in the environment. Unfortunately, we cannot predict and address these problems at design-time. This motivates run-time self-configuration of the localization system.

3.3 Two Forms of Self-Configuration

Since different problems arise at different beacon densities, beacon density should motivate the approach to self-configuration. In Sections 4.1 and 4.2, we discuss the following two forms of self-configuration.

—*At low and medium densities:* Are the deployed beacons enough to guarantee good localization quality throughout the terrain? How do we ensure this? If they are not enough, how can we add beacons to improve the quality of localization.
—*At high densities:* How do we coordinate densely deployed beacons so as to reduce channel contention while best exploiting the spatial diversity and redundancy of densely-deployed beacons?

4. ALGORITHMS

In this Section, we describe the design of HEAP and STROBE, two algorithms for self-configuration for medium and dense beacon deployments respectively.

4.1 HEAP Design

At low and medium densities, beacons deployed in an ad hoc manner for localization may not be sufficient to ensure robust localization throughout the terrain. The goal of HEAP is *incremental beacon placement i.e.,* to discover places to add a few new beacons to maximize improvement in localization, rather than to completely re-deploy the beacon field.

4.1.1 *Overview and Design Rationale.* The HEAP approach to incremental beacon placement is based on system measurements. In HEAP, the wireless network consists of three entities: *Node*, *Beacon* and a *Placer*. A high-level overview of the algorithm is as follows:

—Individual beacons must determine suitable candidate points for adding new beacons within their local neighborhood (for example, within a region of radius r around the

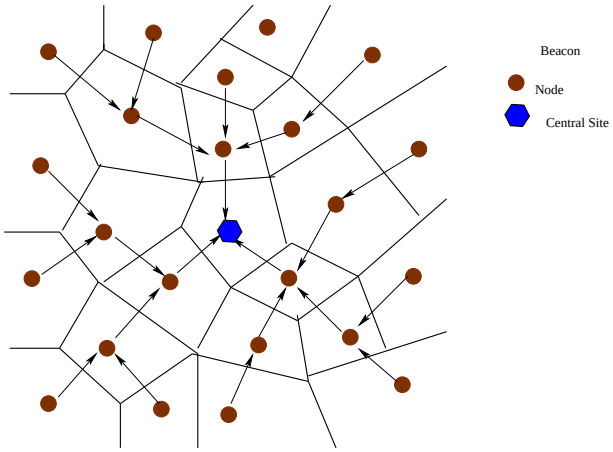


Fig. 2. Information flow for HEAP. Data is transmitted from beacons to the placer, with in network aggregation at intermediate nodes.

beacon). To accomplish this, beacons exchange neighborhood information with each other.

—Because new beacons need to be physically deployed, a controlling agency is needed. The placer deploys new beacons.

—Beacons must send their candidate points to the placer. Intermediate nodes aggregate and relay data from the beacons to the placer. [3]

Information flow in HEAP is illustrated in Figure 2. In HEAP, a central placer is required only because we assume incremental node deployment from a single agency (we selected this definition for comparison with prior central algorithms [Bulusu et al. 2001]). HEAP employs *distributed, in-network* processing to select placement sites. A fully distributed variation on the HEAP algorithm would allow an aggregation node to deploy additional beacons if improvement exceeded some threshold.

4.1.2 *Algorithms.* Several data dissemination mechanisms (LEACH [Heinzelman et al. 2000] and Directed Diffusion[Intanagonwiwat et al. 2000]), and ad hoc routing protocols [Royer and Toh 1999] have been proposed in the research literature.

Information flow in HEAP can be set up using any of these protocols. Once this is in place, the three entities Beacon, Node and Placer execute their parts.

Beacon B : A beacon exchanges information and learns to estimate its beacon neighborhood. It then selects a candidate point and sends it to its $parent$ node.

Node N : An intermediate node in the hierarchy receives candidate points from all its neighbor beacons and child nodes. It selects and forwards one of these candidate points to its $parent$ node.

[3]In systems where the placer is fixed and located far from energy-constrained beacons, hop-by-hop communication rather than direct long range communication to the destination site is preferable for energy-efficiency. Furthermore, it is infeasible to transmit all data across the network, even hop-by-hop. By performing local computation to reduce data before transmission, orders of magnitude energy savings can be obtained [Pottie and Kaiser 2000].

Placer P : The placer receives candidate points from all its neighboring beacons and child nodes. It eliminates any candidate points that do not satisfy constraints and selects good points for adding new beacons.

4.1.3 *Neighborhood Estimation.* Before a beacon in HEAP can select a candidate point, it needs to estimate its beacon neighborhood to the number of hops appropriate to its candidate point selection algorithm. To accomplish this, Beacon B executes the algorithm $BeaconNeighborhood\ (\ B, NumHops\ )$. In this algorithm, beacons include information about other beacons in their neighborhood of a certain scope in their advertisements. A beacon iterates $NumHops$ times, increasing its scope by 1 each time.

Algorithm $BeaconNeighborhood\ (\ B,\ NumHops\ )$
Input: B — A beacon.
$NumHops$ — The Number of hops (or the scope) to which neighborhood must be computed.
Output: A set of all beacons within $NumHops$ of beacon B.

Step 0. $NumPhases \leftarrow NumHops$

Step 1. $Phase \leftarrow 1$

Step 2. $Neighborhood\ (\ 0\ ) \leftarrow position\ (\ B\ )$

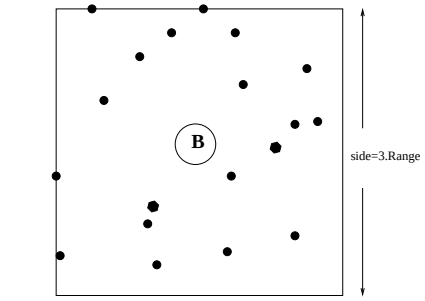
Step 3. while ($Phase \leq NumHops$) do
$BROADCAST\ (\ B,\ Phase - 1,\ Neighborhood\ (\ Phase - 1\ ))$
Listen to broadcasts of other beacons'
$Phase - 1$ neighborhoods.
$Neighborhood\ (\ Phase\ )$ is the union of all the
$Phase - 1$ neighborhoods heard during this period.
$Phase \leftarrow Phase + 1$

Step 4. Return $Neighborhood\ (\ NumHops\ )$

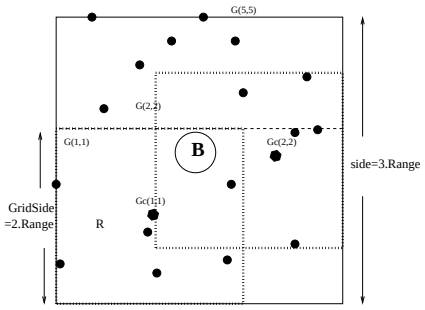
We consider HEAP-GRID, a simple algorithm for selecting candidate points, that extends the basic GRID algorithm proposed in [Bulusu et al. 2001]. We also experimented with HEAP-MAX, the HEAP distributed algorithm with the MAX evaluation function in [Bulusu et al. 2001] (which selects the point with the maximum localization error as a candidate point), but do not report on it here because the HEAP-GRID function gives better improvements in localization quality with respect to the mean and median localization error.

4.1.4 *Candidate Point Selection.* The HEAP-GRID algorithm for candidate point selection, illustrated in Figure 3 learns the neighborhood of a beacon, but with a larger scope of 4 hops. Its approach is to simulate the cumulative localization error over each grid for several uniformly separated points in its neighborhood. It divides the neighborhood into a few square grids, and picks the grid center with the highest error as a candidate point. This is based on the observation that adding a new beacon affects its nearby area, not just the point where it is placed.

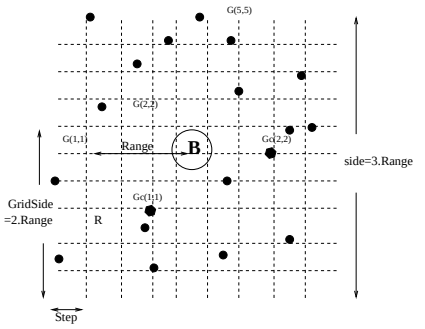
Algorithm $HEAP - GRID(B)$
Input: B - A beacon.
Output: A candidate point where a new beacon could be added.



Step 1



Step 2



Step 3

Fig. 3. Illustration of the HEAP-GRID algorithm. Beacon B determines candidates points in its neighborhood, in this case a square of side $3 \cdot Range$ based on the locations of its neighbor beacons.

Step 0. $NeighborSet \leftarrow BeaconNeighborhood(B, 4)$

Step 1. $side \leftarrow 3 \cdot Range$

$Q \leftarrow position(B) = (X_B, Y_B)$

Consider a square S with side $side$ and center Q.

Step 2. Divide S into N_G partially overlapping grids as follows.

Step 2.1. $gridSide = 2 \cdot Range$

Let the side of each grid be $gridSide$.

Each grid encloses the radio reachability region of its center.

Step 2.2. For $1 \leq i, j \leq \sqrt{N_G}$, the grid $G(i, j)$ is defined by its center $G_c(i, j)$.

$$
\begin{aligned}
G_c(i,j) &= (G_{CX}(i,j), G_{CY}(i,j)) \quad where \\
G_{CX}(i,j) &= X_B - \frac{(side - gridSide)}{2} + \frac{(i-1) \times (side - gridSide)}{\sqrt{N_G} - 1} \\
G_{CY}(i,j) &= Y_B - \frac{(side - gridSide)}{2} + \frac{(j-1) \times (side - gridSide)}{\sqrt{N_G} - 1})
\end{aligned}
$$

Step 3. For each grid $G(i, j)$, compute the cumulative localization error $CE(i, j)$ for the grid $G(i, j)$ as follows.

Step 3.1. Divide the grid into squares of size $step \times step$.

Step 3.2. $\forall\, 0 \leq k, l \leq (\frac{gridSide}{step})$,

let $P(k, l) = (P_X(k, l), P_Y(k, l))$

be the point in the region that corresponds to a square corner.

$$
\begin{aligned}
P_X(k,l) &= G_{CX}(i,j) - gridSide/2 + k \cdot step \\
P_Y(k,l) &= G_{CY}(i,j) - gridSide/2 + l \cdot step
\end{aligned}
$$

Step 3.3. Estimate localization error at each point $P(k, l)$ as follows.

Let χ be the set of all beacons in $NeighborSet$ that are within distance $Range$ of $P(k, l)$.

$LE(P(k, l)) \leftarrow EstimateLocalizationError(P(k, l), \chi)$

Step 3.4.

$$
CE(i,j) \leftarrow \Sigma_{k=0}^{k=\frac{gridSide}{step}} \Sigma_{l=0}^{l=\frac{gridSide}{step}} LE(P(k,l))
$$

Step 4. Return $(G_c(p, q), S(p, q))$ of the grid $G(p, q)$ with maximum cumulative localization error as the selected candidate point.

Although HEAP-GRID is by no means the only possible algorithm, it is representative of the effectiveness attainable with a localized algorithm.

4.1.5 *Error Estimation.* One of the aspects of candidate point selection by a beacon is to estimate localization error at various points based on its knowledge of the beacon neighborhood. This error estimation is the domain-specific part of beacon placement, one can substitute the procedure below for connectivity based localization with other procedures.

Algorithm $EstimateLocalizationError(P, \chi)$

Input: P — a point in 2 dimensional space.

χ — a set of beacons within radio range $Range$ of point P.
Output: An estimate of localization error at point P.

Step 0. $\chi^{'} \leftarrow \{(position(B),\ Range\ ) \mid B \in \chi\ \}$
Step 1. $P_{est} \leftarrow LocalizationFromConnectivity(\chi^{'})$
Step 2. $\epsilon \leftarrow LocalizationError(P, P_{est})$
Step 3. Return ϵ.

4.1.6 *Summary.* We have presented HEAP, our low-complexity algorithm for self-configuration at low and medium beacon densities. HEAP uses the design principle of *localized algorithms*. HEAP is a general framework. The only aspect of HEAP that is domain specific is the error estimation function described in Section 4.1.5.

4.2 STROBE Design

In the previous subsection, we described the HEAP approach. However, a key requirement for large scale sensor networks is robust, unattended operation. Here we would begin with a very dense beacon deployment initially, and then rotate functionality amongst beacons (by turning them on and off) to maximize lifetime. STROBE stands for Selectively TuRning Off BEacons. The goal of the STROBE algorithm is for beacons to cooperatively achieve such an adaptive operational density without diminishing the localization granularity.

4.2.1 *Design Rationale.* For beacon deployment densities μ_{actual} much greater than the saturation threshold μ_{thresh}, tuning the operational beacon density can provide several advantages without diminishing localization granularity. First, the duty cycles of individual beacons can be reduced without diminishing localization granularity, thus increasing system lifetime. Second, with fewer operational beacons at any instant, the overall number of beacon transmissions are reduced, thereby reducing the probability of self-interference amongst beacons. Finally, a higher percentage of beacons could remain active in noisier obstructed parts of the terrain, whereas a smaller percentage of beacons may need to be active in unobstructed terrain, achieving similar localization granularity and the adaptive self-configuration that motivates this work.

We have made some assumptions in the design of STROBE. We state these assumptions and discuss their implications below.

—Beacons are static and compute their position only once. Therefore, we can ignore both the computational and communication energy for continuous position estimation of beacons (for example, GPS acquisition overhead).

—Clients may be mobile and need to update their positions continuously. Therefore, beacons need to remain active throughout the system lifetime. [4] This motivates the need for a continuous or periodically adaptive algorithm like STROBE.

—The interval between successive beacon transmissions remains fixed during the system lifetime. While this is not inherently necessary, it considerably simplifies our design and analysis.

Our goals for STROBE are to: (1) Maintain *uniform localization granularity* across the system and over time. (2) Maximize *system lifetime* by minimizing energy usage at each

[4] If the ratio of clients to beacons is very small, then these systems could be triggered.

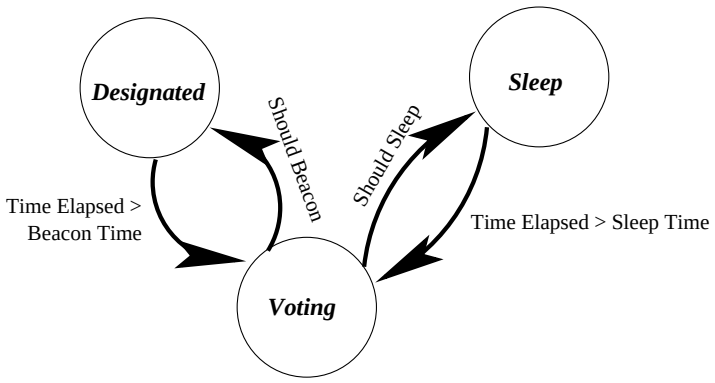


Fig. 4. State Transition Diagram for STROBE. Beacons can switch from Voting to *Designated (D)* or Sleep states and vice versa.

beacon as well as load balancing energy usage across beacons. (3) Minimize *convergence time* of the beacon infrastructure from an initial state to an energy-efficient state. In the initial state, all the beacons are active. In the energy-efficient state, only the threshold level of beacons needed to maintain the desired localization granularity are active. After convergence to a steady state, the system should not deviate significantly from it.

4.2.2 *STROBE Duty Cycle.* Typically, each beacon transmits one position advertisement in a beaconing interval T_B and sleeps for the remainder of the interval. Each position advertisement has four fields:

beacon identifier, beacon position, sequence number, beacon status

Beacon status is usually set to be UP.

In STROBE, a beacon can be in one of three states - *Voting (V)*, *Designated (D)* and *Sleep (SL)*. The state transition diagram is depicted in Figure 4. All beacons start out in the*Voting* state, wherein, a beacon turns on its radio and broadcasts position advertisements every T_B seconds and also listens for advertisements from its neighboring beacons. When a beacon node enters*Voting* state, it sets a timer for T_V seconds. When the timer fires, it evaluates whether it should go to sleep based on a decision making process explained in Section4.2.3. If so, it broadcasts an advertisement with State set to be DOWN and transitions to the Sleep (SL) state. Otherwise, it transitions to the *Designated (D)* state. A beacon node in sleep state wakes up after a sleep time T_{SL} and transitions back to*Voting (V)* state. A beacon node in *Designated (D)* state periodically advertises at intervals T_B for a time T_D and then transitions back to*Voting (V)* state. A beacon node in sleep state wakes up after a sleep time T_{SL} and transitions back to*Voting (V)* state.

Distinct*Voting* and *Designated (D)* states are necessary in order to avoid the overhead incurred due to receiving advertisement messages from other neighbor beacons when in the Voting state. Three important parameters of STROBE that influence its energy usage and system lifetime are T_V, T_D, and T_{SL}.

4.2.3 *Beacon Decision Making.* During the *Voting (V)* state, a beacon evaluates ζ, the number of currently active beacons that are its neighbors.

$$\zeta = |B_{up} - B_{down}| \tag{12}$$

where B_{up} is the set of all beacons it heard from whose most recent advertised state is UP and B_{down} those whose most recent advertised state is DOWN.

This means that the number of active beacons in its neighborhood, including itself is $\zeta + 1$. Let μ_{thresh} be the threshold number of beacons in any given neighborhood at which the localization granularity saturates.

If $(\zeta+1) \leq \mu_{thresh}$, then it has to remain active. If $(\zeta+1) > \mu_{thresh}$, then its transition probability p to the *Designated* state is given by:

$$p = \frac{(\mu_{thresh} - 1)}{\zeta} \tag{13}$$

With probability $(1 - p)$ it transitions to the *Sleep* state.

Note that this is a very simple decision making approach, influenced only by the number of currently active neighbors ζ.

More sophisticated approaches could incorporate information such as energy reserve of a beacon and its neighbors, as well as bias a beacon's current estimate of ζ based on a previous history of measurements. However this would require beacons to maintain some additional state.

4.2.4 *Summary.* We have presented STROBE, our algorithm for self-configuration at high beacon densities. The quality of proximity-based localization saturates at a certain beacon density. STROBE builds on this observation to rotate functionality amongst redundant beacons and extend system lifetime. We described the duty cycle of beacons in STROBE, and its decision making approach. We presented our justification for choosing three states in STROBE and the relative time periods for each state via energy usage analysis. Like HEAP, STROBE is also a general approach. The only aspect of STROBE that is domain specific is the decision making function.

5. EVALUATION

We have presented the design of two algorithms, HEAP and STROBE. In this Section, we evaluate these algorithms using both simulations and experiment and discuss the implications of our findings. We use the Berkeley Rene mote testbed for our experiments, described in Appendix A.2. Our measurement techniques and tools are described in Appendix B.

5.1 HEAP Evaluation

In this subsection, we report on some results from a performance evaluation of our HEAP algorithm.

5.1.1 *Simulations.* We use simulations to explore, in some detail, the implications of several design choices.

Goals, Metrics and Methodology. Our goals in conducting this evaluation were two-fold: (i) Compare HEAP-GRID performance to a completely Random algorithm as well as to a centralized algorithm (GRID) with global knowledge of beacon positions and terrain or connectivity conditions. (ii) Understand the impact of noise caused by propagation losses and terrain features on the beacon placement algorithms.

We choose the following two metrics to analyze the performance of our algorithms. These metrics are statistics evaluated by sampling the localization error at all $step \times step$

Table II. Terrain-influenced Shadowing Model Parameters

PARAMETER	DEFINITION	VALUE
λ	Wavelength	0.333m
β_1	Path loss exponent(unobstructed)	2
β_2	Path loss exponent (obstructed)	4
σ_{dB}	Standard deviation of noise	5
P_t	Transmitted Power	660mW

square corners obtained by subdividing the region.

(1) *Improvement in mean localization error* M_1 computes the difference between mean localization error at all measured points in the terrain before and after the beacon node is added. This metric indicates the overall impact of adding a beacon to quality of localization in the entire terrain. For a given beacon placement $\mathcal{BP}$, recall that

$$\textit{MeanErr}(\mathcal{BP}) \;=\; \frac{\Sigma_{k=0}^{\frac{Side}{step}} \Sigma_{l=0}^{\frac{Side}{step}} LE_{\mathcal{BP}}(P(k,l))}{(\frac{Side}{step}+1)^2}$$

where $P(k,l) = (k \cdot step, l \cdot step)$

$$M_1 \;=\; \textit{MeanErr}(\mathcal{BP}_{init}) - \textit{MeanErr}(\mathcal{BP}_{final}) \tag{14}$$

(2) *Improvement in median error* M_2 computes the difference between the median localization error at all the measured points in the terrain before and after the beacon node is added. This metric indicates the improvement due to adding a beacon on the quality of localization at the top 50% of the points with the highest localization error at the terrain.

$$M_2 \;=\; \textit{MedianErr}(\mathcal{BP}_{init}) - \textit{MedianErr}(\mathcal{BP}_{final}) \tag{15}$$

We study these metrics as a function of beacon density. In addition, we assume $step = 1m$.

To understand how HEAP copes with noisy radio propagation, we evaluated HEAP for both (i) ideal radio propagation conditions and (ii) a terrain based shadowing model (uses a bitmap of the terrain). We ported the latter from Arena/ns [Ye et al. 2001] to our simulations. The experiments were carried out in a simulated square terrain of side 100m.

From Figure 6 we can see that the environment contains both obstructions and good terrain, so the terrain based propagation model is quite appropriate. The various propagation model parameters we chose is summarized in Table II. The values of β and σ_{dB} are chosen from the ranges of their typical values [Rappaport 1996]. The terrain-based shadowing model has different values of β for line of sight and non line of sight respectively. P_t, the transmit power is selected from [Kaiser 2000] and P_{thresh}, the receiving threshold is set to be the receive power at the nominal radio range $Range$ using Friis free space model [Rappaport 1996]. These do not necessarily reflect the details of real environments, but are representative of a range of environments in which our algorithms may be used.

Impact of beacon density. To compare the performance of HEAP, our localized algorithm for various beacon densities with GRID, the centralized measurement based algorithm described in [Bulusu et al. 2001], we conducted the following simulation experiment. We varied the number of beacons, N from 20 to 80 in increments of 20. The nominal radio transmission range of a beacon $R = 15m$. Correspondingly, μ, the number of beacons per nominal radio coverage area ($bpnrca$) varies from 1.41 to 5.64.

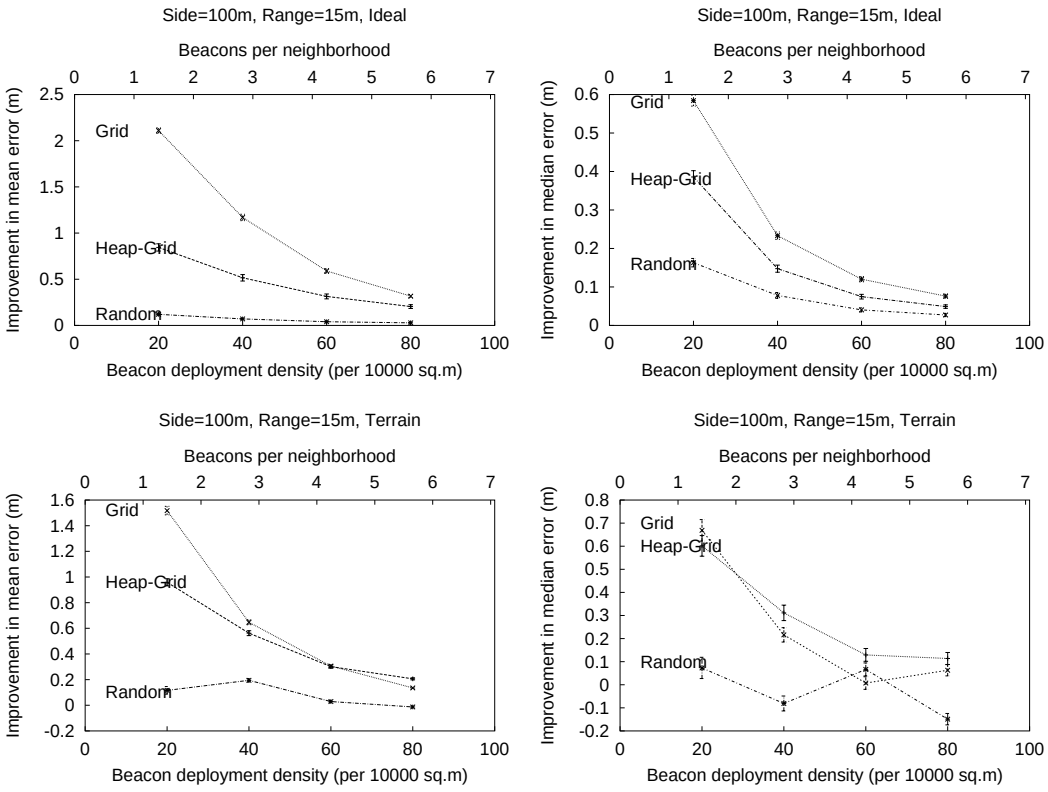


Fig. 5. Performance comparison of HEAP with centralized algorithms for the mean and median localization granularity metrics vs. density.

We generated 1000 different beacon fields per beacon density. Each beacon field is generated by randomly placing the beacons in the $100m \times 100m$ square terrain. Performance metrics for each algorithm and beacon density are averaged over the 1000 beacon fields. To characterize the stability of our results, all graphs include 95 percentile confidence intervals.

Figure 5 plots the improvements in the mean and median localization errors as a function of beacon deployment density for both ideal radio propagation model and the terrain based shadowing model.

With ideal radio propagation, both algorithms perform well for low densities ($< 3\ bpnrca$), but GRID has the potential for significant improvements. For all the three algorithms, the metric M_1 (improvement in mean localization error) decreases rapidly for densities $\geq 3\ bpnrca$, and saturates for densities $\geq 6\ bpnrca$. The gain in median localization error (metric M_2), for GRID relative to HEAP-GRID is considerably lower than metric M_1. Because HEAP-GRID selects candidate points only in the local neighborhood, it is unlikely to identify noisy points as well as the centralized algorithm does. Its worst case improvement, and consequently, mean improvement M_1 tends to be much smaller.

The trend exhibited by metric M_2 for ideal radio propagation is further exemplified for the terrain-influenced shadowing model for radio propagation, as Figure 5 shows. In the

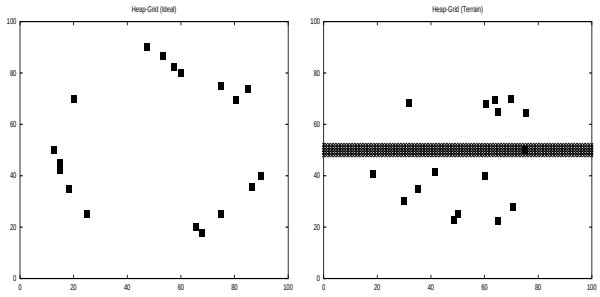


Fig. 6. Candidate points selected in 20 runs of HEAP uniform placement for the ideal case and for a terrain with a wall in the middle. Candidate points shift closer to the center when there is a wall in the middle.

terrain case, for low densities, the total number of noisy points far exceeds the ideal case. GRID which instruments the whole terrain leverages this and posts higher gains in mean error by substantially improving a large number of bad points. HEAP-GRID focuses on moderately bad points and therefore improvement is relatively lower. The median error improvements for the terrain case for HEAP are also much better for higher densities.

Although the gain for HEAP does not equal the centralized algorithm, both are comparable. Moreover, HEAP is distributed and therefore much more scalable.

Impact of terrain features. To qualitatively evaluate the effectiveness of HEAP in selecting good candidate points in a noisy terrain, we conducted a second simulation experiment wherein initial beacon placement is always uniform, varying the number of beacons N and the transmission range R. N = 25, 36, 49, 64, 81 and 100. R = 15m, 20m, and 25m. In each case, HEAP is run to determine the candidate points for two scenarios (a) an ideal terrain with no obstacles and (b) a terrain with a wall in the middle shown in Figure 6. In Figure 6, each point represents a new placed beacon from one simulation run. The right plot adds a wall (shown in grey) as terrain. A simple boundary constraint is applied to remove algorithm bias towards candidate points at the corners of the terrain.

In the ideal case, HEAP-GRID selects candidate points closer to the periphery of the region enclosed by the boundary constraint. This is because it selects the center of the grid with the highest cumulative localization error, and in the ideal case such grids are more likely to be located at the edges of the terrain (even with uniform beacon placement and the boundary constraint). For the terrain, the candidate points shift closer to the center near the wall. The actual points selected depend on the beacon density, range and positions of the beacons relative to the wall.

Despite having to deal with erroneous information (poor neighborhood approximation, idealized radio model etc.), the HEAP algorithms are able to select candidate points closer to a terrain feature such as a wall. However, such a result may not be valid for very small terrain objects, such as foliage.

5.1.2 *Experimental Results.* We also evaluated HEAP in a real environment to verify its effectiveness. We deployed a localization system consisting of 16 beacons in an indoor environment (see Figure 7), in the configuration shown in Figure 8. We chose an indoor setting for this experiment because the radio propagation is not ideal indoors due to multipath effects, and therefore it provides us an interesting test case to study how well HEAP

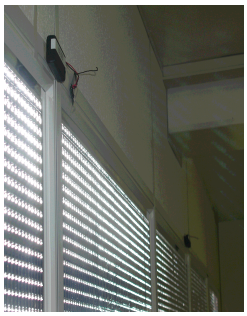


Fig. 7. Beacon deployment in the UCLA Laboratory for Embedded Collaborative Systems (LECS). Motes are attached to the ceiling tiles.

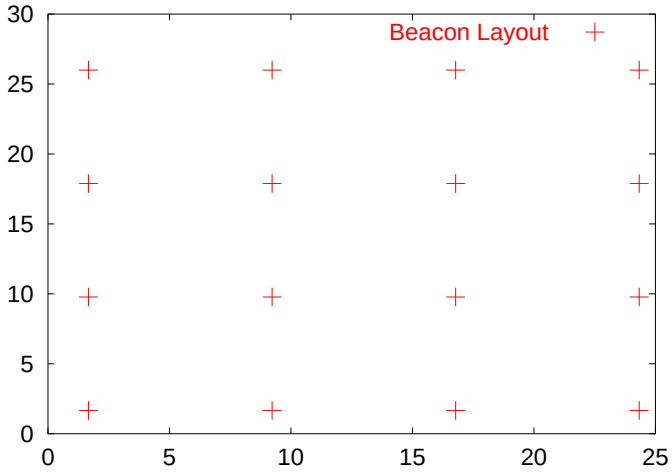


Fig. 8. The configuration in which beacons are placed in the LECS laboratory. Each + sign indicates a beacon. 16 beacons are uniformly located in a 24×24 $feet$ square region, with adjacent beacons 8 $feet$ apart.

helps the system adapt to its environmental conditions.

We varied the transmission power and frequency settings using software-enabled control commands (see Appendix B). For each unique setting, we collected the following data:

—*Beacon Connectivity Measurements:* Each beacon measures its connectivity to other beacons. We obtain the beacon network topology from the connectivity measurements of all beacons.

—*Localization Error Measurements:* We measure localization error at various points in the terrain by walking across the room and collecting data at spacings of 2 feet. For each point, the localization error is averaged over several trials.

We found real experiments to be very valuable. We observed the following for our experiment with 16 beacons:

—At the same physical point, the localization estimate varied over time.

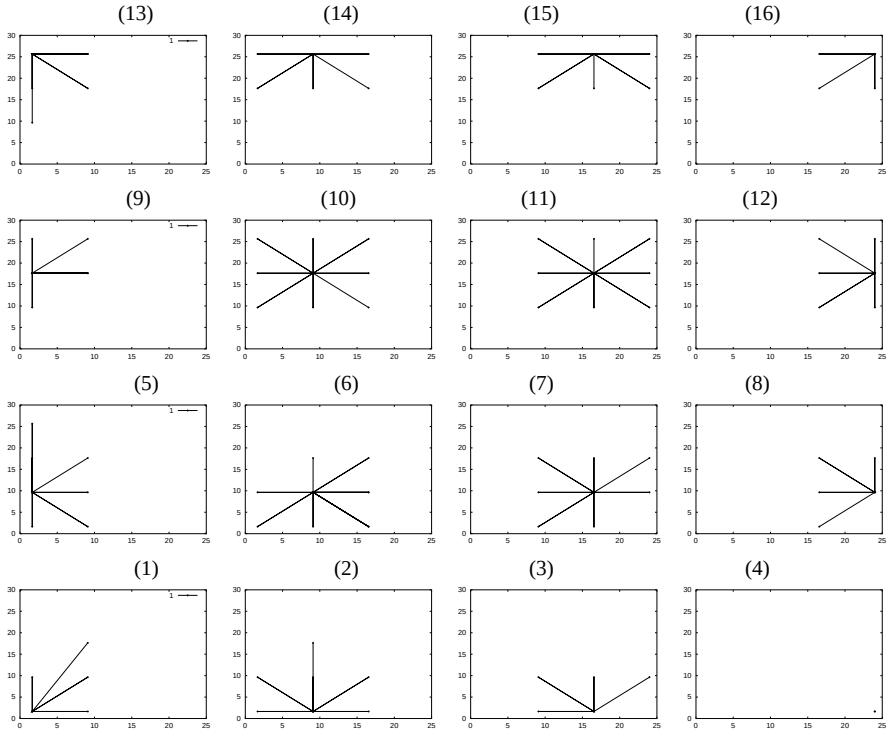


Fig. 9. Beacon Connectivity Graph in our experiment. Connectivity of each beacon is shown in a separate graph and beacons are numbered. Connectivity between beacons is sometimes asymmetric (as in beacons 1 and 6), and some beacons have a greater connectivity degree than others (compare 5 with 9). The corner beacon 4 has no connectivity.

Table III. Control Parameters for the Beacon System

PARAMETER	VALUE
Transmission Power/Potentiometer Setting	75
Beaconing Interval (seconds)	3

—The connectivity relation between two beacons varies over time.

Because candidate point selection in HEAP is based on beacon connectivity relations, the connectivity graph provides us complete information to emulate the HEAP algorithm. We deployed new beacons at candidate points selected by HEAP and recomputed the localization error at various points in the terrain.

Table III refers to the control settings used for the experiment whose results are shown in Figures 9 and 10.

Figure 9 plots the Beacon Connectivity Graph. We can see that it is asymmetric, some beacons have greater connectivity than the others. This connectivity graph was used to emulate the HEAP algorithm.

Figure 10 displays the candidate point selected by HEAP to add a new beacon. We can see that the candidate point is very close to the position of the failed beacon (beacon 4 in Figure 9).

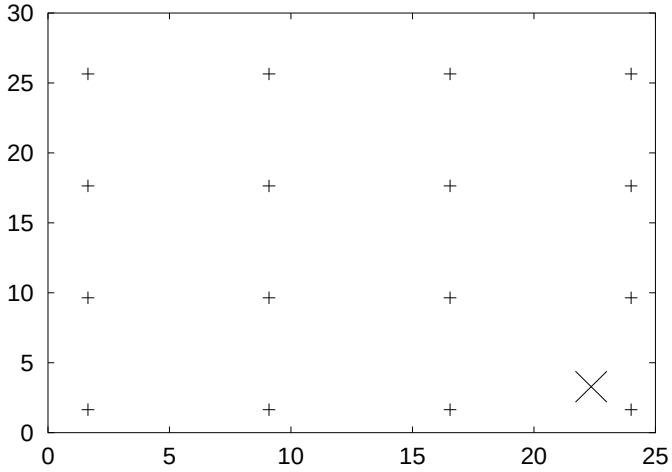


Fig. 10. Candidate Point Selected by HEAP. The plus (+) sign indicate positions of beacons. The cross (X) sign indicates the position of the candidate point selected by the HEAP algorithm.

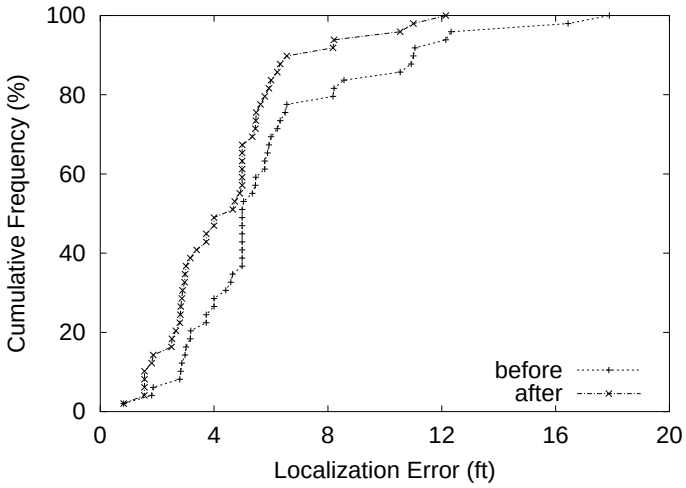


Fig. 11. A comparison of cumulative distribution function (CDF) of localization error before and after the beacon is added at the candidate point selected by the HEAP algorithm.

We added a new beacon at the candidate point. Figure 11 plots the cumulative distribution function of the localization error before and after the new beacon was added. While the median error remains the same, there is significant improvement in the 90 percentile error (drops almost 50% from 11 $feet$ to 6 $feet$). This shows us that HEAP can be effective in a real environment.

5.1.3 *Discussion.* From our design and evaluation of HEAP, we can draw the following general lessons.

(1) Our simulations show that localized and adaptive algorithms such as HEAP are effective in comparison to centralized adaptive algorithms such as GRID in addressing beacon placement. This is because the relevance of information needed by a beacon for algorithmic computation drops as a function of distance or number of hops to the beacon.
(2) Our experimental results show that HEAP can benefit a real deployed localization system. This proves that adaptive self-configuration to terrain and environment characteristics based only on local coordination among beacons and without a system or terrain model is both feasible and worthwhile.
(3) Proximity-based localization has a beacon density beyond which the benefit of additional beacons falls off. This observation suggests the STROBE algorithm targeting high beacon densities, evaluated next. More generally, the study of performance as a function of density is important for algorithms involving many nodes.

5.1.4 *Summary.* We presented detailed simulations to show that HEAP can achieve results comparable to centralized adaptive algorithms. We presented experimental results that demonstrated the benefits of HEAP in a real deployed localization system.

5.2 STROBE Evaluation

We have also conducted extensive evaluations of STROBE. We discuss our findings here.

5.2.1 *Simulation Results.*

Goals, Metrics and Methodology. Our goals in evaluating STROBE using simulations are to answer the following questions:

—Is STROBE effective?
—How do various parameters affect its performance?
—How well does STROBE perform compared to the optimal case?

We use several metrics in our evaluation. We study the following metrics as a function of time.

— *% Beacons Active $P_{active}(t)$:* Percentage of total beacons that are in either *Voting (V)* or emph Designated (D) states at any given instant of time.
— *% Beacons Alive $P_{alive}(t)$:* Percentage of total beacons that possess energy reserves greater than zero at any given instant of time.
— *Median localization error of the terrain $MedErr(t)$:* at any given instant t can be calculated by substituting $\mathcal{BP}_{active}(t)$ for $\mathcal{BP}$ in Equation 6, where $\mathcal{BP}_{active}(t)$ is the placement of the set of beacons active at time t and $step = 1m$.

We use two other metrics.

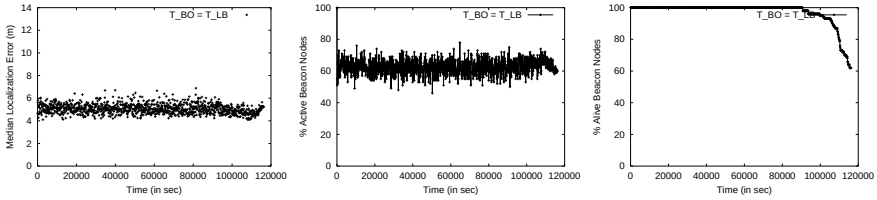
— *First node death:* Time elapsed since the start before any single node in the terrain runs out of energy (dies).
— *System lifetime:* Time elapsed since the start before the median localization error exceeds a n operational error threshold. (for example: $0.4 \cdot Range$)

For our simulations, we choose an energy consumption model to mimic realistic sensor radios [Kaiser 2000]. These parameters are also used in [Intanagonwiwat et al. 2000] and are summarized in Table IV.

Table IV. Energy Consumption Parameters Used in STROBE Evaluation.

POWER DISSIPATION	RADIO OPERATION MODE	VALUE
P_X	Transmit	660 mW
P_R	Receive	395 mW
P_I	Idle	35 mW
P_S	Sleep	0 mW

$T_D = T_V$



$T_D = 100 \cdot T_V$

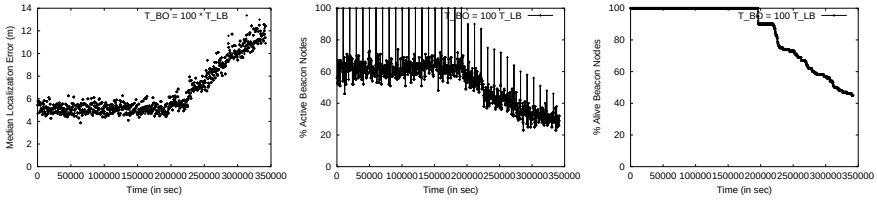


Fig. 12. STROBE performance for two ratios of $\frac{T_D}{T_V}$. Top Row $\frac{T_D}{T_V} = 1$, Bottom Row $\frac{T_D}{T_V} = 100$. $R = 20m$, $N = 100$, T_B = 1s, $T_V = 5T_B$, Φ = 10000J, Snapshot period = 100s.

Sensitivity to STROBE parameters. To study the sensitivity of STROBE performance to its parameters, especially the rate of adaptation, $\frac{T_V}{T_D}$, we simulated a terrain of area $100m \times 100m$ with 100 randomly placed beacons in the terrain. The nominal radio range is $20m$. Thus the number of beacons per nominal radio coverage area is around 12 ($\mu_{actual} = 12, \mu_{thresh} = 6$). Each node has a starting energy of 10000J. Transmit time (T_X) of a beacon advertisement is 0.025 seconds. Beaconing interval T_B is set to be 1 second. T_V is set to be 5 seconds and T_D is varied to be T_V and $100T_V$.

Figure 12 compares the performance of the STROBE algorithm for various ratios of $\frac{T_D}{T_V}$ with respect to these metrics: median localization error, percentage of active beacons, percentage of beacons alive at nodes. The simulation terminates when none of the nodes has sufficient energy to either transmit or receive packets.

Increasing the ratio $\frac{T_D}{T_V}$ improves the system lifetime. For instance, the first node deaths occur at 90000 seconds and 200000 seconds respectively for values of $\frac{T_D}{T_V}$ set to 1 and 100 (heuristically chosen). It also improves the time duration between the first node death t_F and the last node death t_L. In addition it also minimizes the variations in median error over small periods of time.

The median localization error over time is closely correlated to the percentage of beacons alive. The step wise degradation (*i.e.,* increase) in the median localization error after the first node death mirrors the step wise decrease in the percentage of beacons alive over time.

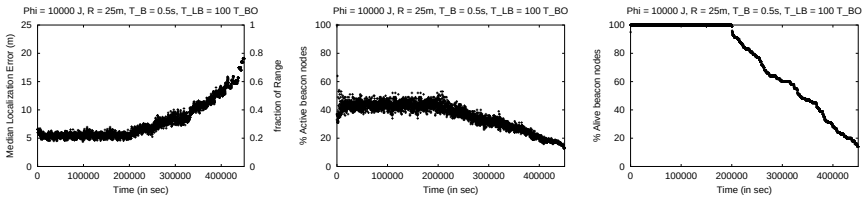


Fig. 13. STROBE performance for N=100, R=25m, $T_B = 0.5s$, $T_V = 2T_B$, $T_D = 100T_V$, Φ=10000J.Lifetime of the algorithm using simple beaconing $L_B = 300000s$. Snapshot time = 100s.

A closer inspection of the terrain snapshots over time reveals that because beacons are distributed uniformly in limited-size terrain, we see boundary conditions at the edges. For boundary beacons, the observed neighborhood size is either close to or less than μ_{thresh}, therefore they all tend to remain active and die first at approximately the same time. The next phase occurs when the next set of beacons that die are the ones that were adjoining the previous boundary beacons and are now the new boundary beacons, leading to a cascading failure of nodes.

STROBE Benefits. Our second simulation experiment demonstrates STROBE benefits for an applicable context (small beaconing interval, high beacon density). We simulate a terrain with 100 beacons distributed uniformly at random in a $100m \times 100m$ terrain. The nominal radio range of these beacons is 25m. The corresponding beacons per neighborhood μ_{actual} = 19 = 3.1μ_{thresh}. We choose a reasonably small beaconing interval, $T_B = 0.5$ seconds. We set the various STROBE parameters as follows: $T_B = 0.5s$, $T_V = 2T_B$, $T_D = 100T_V$, Φ=10000J. The lifetime of a beacon using simple beaconing L_B is

$$L_B = \frac{\Phi}{T_B} \tag{16}$$

In this case, $L_B = 300000s$.

We omit our detailed analysis of STROBE energy usage due to space restrictions, but the best case system lifetime in STROBE can be shown to be

$$\mathcal{L}_{STROBE} = \frac{\mu_{actual} L_B}{\mu_{actual} \frac{\frac{P_V}{P_D} - 1}{2 + \frac{T_D}{T_V}} + \mu_{thresh}} \tag{17}$$

where μ_{actual} is the actual number of beacons per neighborhood, μ_{thresh} is the threshold number of beacons per neighborhood for localization, P_V and P_D are the mean power dissipated in the *Voting* and *Designated* states respectively.

Figure 13 plots the median localization error, percentage of active beacons and percentage of beacons alive as a function of time. Snapshots are taken every 100 seconds. The degradation in median localization error as well as percentage of beacons alive over time is considerably smoother than in our previous simulation experiment. In this case, STROBE maintains a median localization error within $0.2 \times Range$ for up to 200000 seconds, $0.3 \times Range$ for up to 300000 seconds, and $0.5 \times Range$ for up to 400000 seconds. Actual system lifetime (L_{STROBE}) is increased to around 450000 seconds or

$1.5L_B$. This is low compared to the best case lifetime predicted by our model substituting $\frac{T_D}{T_V} = 100$ of 850000 seconds $2.8L_B$. That calculation assumes energy usage can be load balanced effectively across beacons and that beacons are uniformly distributed in the terrain. However, as we have seen boundary nodes tend to die first, causing a cascading effect. To improve further on these lifetimes, beacons could perform edge detection to identify boundary conditions and adjust their beaconing period T_B to be higher compared to other beacons. Alternatively, a higher density of beacons could be deployed near the boundary.

STROBE transitions probabilistically from *Voting (V)* to *Sleep (SL)* states, causing a higher percentage of beacons than the threshold percentage to remain active. Leveraging auxiliary information may significantly improve this lifetime.

5.2.2 *Experimental Results.* We have also tried to evaluate STROBE experimentally. This is slightly harder to do because we have to measure both the energy depletion at different nodes over time and the degradation of localization quality at various points across the terrain and over time (which requires manual intervention and is therefore not feasible at a very fine grained time scale). Instead, the methodology we used was experimental emulation.

—We collect real beacon connectivity data and play back this connectivity data in a custom simulator to emulate the beacons' decision making process in STROBE. In modeling the behavior of a localization system, radio propagation is the hardest to model well, and hence it is important to verify it using real data.

—Simulate power consumption over time using a radio energy model. Since radio communication dominates the power consumption of these nodes (as opposed to computation), this provides us with a good approximation of energy usage. Moreover, by using the same energy consumption model as our simulation, we can also validate the simulation.

—Emulate localization error in our connectivity based localization method using the connectivity data. This allows us to analyze the degradation in localization quality at a very fine-grained time scale.

Figure 14 plots the median localization error as a function of time (for both the experiment and simulation). We notice that the system lifetime with experimental emulation is comparable to idealized simulation, but the quality of localization is only slightly worse (20%). Thus, our idealized simulations can be considered a good indicator of STROBE performance. The localization quality is slightly worse in the experimental case because we did not account for link asymmetry in initial design of the STROBE decision making. Beacons can turn themselves off even when neighbor is a stray far-away beacon. To avoid this, we may need a geographic filtering technique in the beacon decision making process.

5.2.3 *Discussion.* We can draw two general lessons from our design and evaluation of STROBE.

(1) For density regimes above the threshold density, our example shows that a completely localized algorithm like STROBE can extend the system lifetime 1.5 times without diminishing localization granularity with 3.1 times saturation density of nodes. Lifetime gains can be improved further for higher beacon densities and energy dissipation rates in active state, and by augmenting STROBE with boundary detection mechanisms.

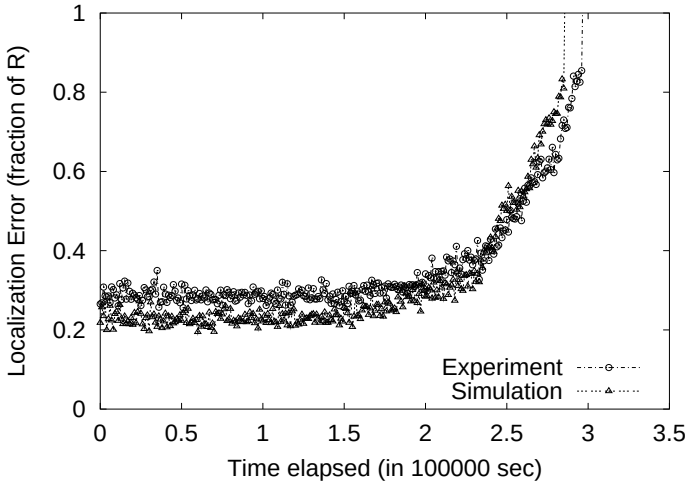


Fig. 14. A comparison of median localization error vs. time for the experimental emulation with the simulation

(2) Adaptation to terrain conditions and node availability invariably has an associated measurement overhead. Therefore adaptive density should be applied only when the benefit of adaptation greatly exceeds its overhead. Examples of this are high density beacon deployment and high energy dissipation in active states. STROBE is not justifiable in contexts when beacons are already operating at a very low duty cycle or when the deployment density is not high enough to provide enough interchangeable beacons.

5.2.4 *Summary.* We presented detailed simulations to show that STROBE converges quickly, maintains uniform localization quality in the terrain, and over time, and can significantly extend overall system lifetime. We also presented experimental results that validate our simulation methodology.

5.3 Discussion

We have presented the design and evaluation of HEAP and STROBE. We now discuss limitations and lessons from our experimental evaluation and issues for future design.

5.3.1 *Experimental Evaluation: Limitations and Lessons.* Although the motes used in our experiment are largely automatically manufactured, they are not identically calibrated. Motes from different shipments produced different levels of noise (2 KHz and 20 KHz). Moreover, the antennas are handmade, and the reference voltage which is obtained from a battery source is not stabilized. These are largely responsible for asymmetric radio connectivity effects. Sensor self-calibration is currently the focus of research efforts at UCLA and Berkeley.

A dominant issue in sensor network design is power conservation. While we have measured real radio data, we have not measured real power consumption. Instead, we have emulated energy-consumption based on the radio communications energy expended (but have ignored computation energy). It is increasingly desirable to evaluate these algorithms using real power measurements.

Self-configuring systems are challenging to measure and analyze because the system is constantly in flux and never in steady state. Our experience suggests that while measurements are invariably difficult to make, they are extremely necessary. For instance, our design (and simulations) of HEAP and STROBE did not account for asymmetric connectivity, but our experiments revealed that it does have an impact on the number of beacons that decide to stay active.

5.3.2 *Future Design Issues.* Both HEAP and STROBE can be easily generalized to other localization systems using beacons, when the position estimation is not based on proximity. In HEAP, only the error-estimation part is domain specific, and will need domain-specific information. In STROBE, only the decision making step described in Section 4.2.3 needs to change.

We have designed and evaluated several variants of HEAP. A complete discussion of the variants of HEAP is out of scope for this paper. The first of these variants extends HEAP to add multiple beacons, not just one. In the second variant, we substitute error-estimation function with a geometry-aware one, more suitable when position is a function of range measurements. Neighborhood discovery in HEAP is accomplished through a multi-hop protocol. We can also accomplish this by transmitting advertisements at a higher power.

In STROBE, the decision of a beacon to remain active or sleep is influenced only by requirements to maintain a uniform localization granularity across the system at all times. We may not really need homogeneous localization granularity in the system at all times, especially if the system is event based [Schurgers et al. 2002]. Instead, we may want the system to self-configure in response to application dynamics or events. We have experimented with triggered beacon systems in our laboratory and these can lead to orders to magnitude improvements in energy-conservation.

Finally, a comment on the quality of localization. While a saturation threshold of $0.25Range$ may seem too high, the use of multiple power-levels can significantly improve localization granularity. Additionally, they can be exploited to improve decision making of the beacons, especially when asymmetry exists in beacon connectivity.

While the localization quality can be improved by using multiple sensor modalities or range measurements [Girod and Estrin 2001] instead of radio proximity, these will not completely eliminate the need for self-configuration. Better algorithms may diminish error or eliminate outliers, but we need enough redundancy in measurements to employ filtering techniques. And self-configuration will be needed to ensure that redundancy. Thus our work complements these techniques.

Lately there has been significant research in self-configuring network protocols for large, dense ad hoc wireless networks - to form network topologies [Cerpa and Estrin 2002], routing [Xu et al. 2001], media access [Ye et al. 2002] and beacon systems. In sensor networks, nodes may be participating in many self-configuring tasks at different network layers. Integrated self-configuration that takes into account all factors is an issue for future research. Additionally, the performance of these protocols is extremely sensitive to the choice of network parameters. Consequently, it is important to establish a theoretical foundation for self-configuring systems. Some initial progress in this direction has been made by [Krishnamachari et al. 2002].

6. CONCLUSIONS

Large-scale, densely distributed sensor networks that are closely coupled to the physical world require node localization, but under far severe node-level resource constraints (limited energy, bandwidth, memory and processing) [Bulusu et al. 2000]. Existing geolocation systems such as GPS (the Global Positioning System) do not always meet their operational (low power), environmental (indoors) or cost constraints. Localization systems that can reconcile these needs by necessity must be loosely coupled, distributed systems [Bulusu et al. 2000] [Priyantha et al. 2000] [Savvides et al. 2001] [Girod 2000] [Hightower et al. 2000].

We have highlighted the deployment, configuration and operational issues of such a localization system and argued that it must itself *self-configure*, that is, autonomously measure and adapt to the environmental and system dynamics in order to achieve environmental independence and robust, unattended system-level operation. In this paper, we have presented the design and evaluation of algorithms to achieve that self-configuration. Our design process relied on the following three pieces of insight.

First, beacons are one key approach to localization as they can guarantee the convergence and accuracy of loosely-coupled distributed localization systems (see [Savvides et al. 2001]). The localization granularity can be directly related to the beacon density when beacons are distributed uniformly at random. Our simulations show that the localization granularity saturates at a threshold beacon density μ_{thresh}. This realization argues for a beacon density-sensitive approach to self-configuration.

Second, the beacon density is not a homogeneous phenomenon in real environments. Rather, the beacon density varies throughout the terrain due to deployment perturbations and due to environment-dependent propagation vagaries even when beacons are placed uniformly. Note that this observation implies that just deploying beacons below or at this theoretical density μ_{thresh} will not be adequate. Instead, beacons must themselves establish the density through measurements and suggest local candidate points where new beacons could be added so as to improve the localization quality, as in the HEAP algorithm proposed herein. Care was taken to propagate neighborhood information beyond a single hop, so that beacons can select the candidate points effectively.

Third, beacons contend for the wireless channel when they broadcast advertisement packets containing their position. When beacons are deployed at high densities (greater than μ_{thresh}) in order to provide redundancy, the responsiveness and granularity of the system degrades due to the self-interference caused by the channel contention. Instead of having all the beacons simultaneously participate, beacons must explicitly coordinate so that only some of them participate at a time. For a variety of performance reasons, characterizing the measured and threshold density allowed a simple randomized algorithm that achieved the desired statistical behavior in maintaining localization granularity. Careful analysis of energy usage allows us to tune sleep probabilities and periods so as to maximize system lifetime. This idea could be relevant to not only beacons but also to routing [Xu et al. 2001], media access protocols[Ye et al. 2002] and topology control [Cerpa and Estrin 2002].

Our experimental results show that these various algorithms have significantly improved the performance of the localization system proposed in [Bulusu et al. 2000]. We discussed in Section 5.3, the limitations of our evaluation and design issues for the future. Now that the we have validated our self-configuring localization methodology experimentally, we look forward to deploying it in our applications.

REFERENCES

AURENHAMMER, F. 1991. Voronoi diagrams - a survey of a fundamental geometric data structure. *ACM Computing Surveys 23*, 345–405.

BAHL, P. AND PADMANABHAN, V. N. 2000. Radar: An in-building user location and tracking system. In *Proc. of IEEE Infocom 2000*. Vol. 2. IEEE, 775–84.

BULUSU, N., HEIDEMANN, J., AND ESTRIN, D. 2000. Gps-less low cost outdoor localization for very small devices. *IEEE Personal Communications Magazine 7*, 5 (October), 28–34.

BULUSU, N., HEIDEMANN, J., AND ESTRIN, D. 2001. Adaptive beacon placement. In *Proc. of ICDCS-21*. IEEE Computer Society, Phoenix, Arizona, USA, 489–498.

CERPA, A. AND ESTRIN, D. 2002. Ascent: Adaptive self-configuring network topologies. In *Proc. of IEEE Infocom 2002*. Vol. 2. IEEE, New York, USA.

CHARIKAR, M., GUHA, S., SHMOYS, D., AND TARDOS, E. 1999. A constant-factor approximation algorithm for the k median problem. In *Proc. of ACM STOC 1999*. ACM, 1–10.

COX, I. J. 1991. Blanche - an experiment in guidance and navigation of an autonomous robot vehicle. *IEEE Transactions on Robotics and Automation 7*, 2, 193–204.

DOHERTY, L., PISTER, K. S., AND GHAOUI, L. E. 2001. Convex position estimation in wireless sensor networks. In *Proc. of IEEE Infocom 2001*. Vol. 3. IEEE, Anchorage, Alaska, 1655–1663.

ESTRIN, D., GOVINDAN, R., HEIDEMANN, J., AND KUMAR, S. 1999. Next century challenges: Scalable coordination in sensor networks. In *Proc. of ACM MOBICOM '99*. ACM, Seattle, WA, USA, 263–270.

GIROD, L. 2000. Development and characterization of an acoustic rangefinder. Tech. Rep. USC-CS-TR-00-728, University of Southern California. April.

GIROD, L. AND ESTRIN, D. 2001. Robust range estimation for localization in ad hoc sensor networks. http://lecs.cs.ucla.edu/ girod/papers/NLOS.ps.

GUIBAS, L., LIN, D., LATOMBE, J. C., LAVALLE, S., AND MOTWANI, R. 2000. Visibility-based pursuit evasion in a polygonal environment. *International Journal of Computational Geometry Applications*.

HEINZELMAN, W., CHANDRAKASAN, A., AND BALAKRISHNAN, H. 2000. Energy-efficient communication protocols for wireless sensor networks. In *Proceedings of the HICSS*.

HIGHTOWER, J., WANT, R., AND BORRIELLO, G. 2000. Spoton: An indoor 3d location sensing technology based on rf signal strength. UW CSE 2000-02-02, University of Washington, Seattle, WA. February.

HILL, J., SZEWCZYK, R., WOO, A., HOLLAR, S., CULLER, D., AND PISTER, K. 2000. System architecture directions for networked sensors. In *Proc. of ASPLOS-IX*. ACM., Cambridge, MA, USA, 93–104.

HOFMANN-WELLENHOFF, B., LICHTENEGGER, H., AND COLLINS, J. 1997. *Global Positioning System: Theory and Practice, Fourth Edition*. Springer Verlag.

INTANAGONWIWAT, C., GOVINDAN, R., AND ESTRIN, D. 2000. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proc. of MobiCom 2000*. ACM, N.Y.

KAISER, W. J. 2000. Wins ng 1.0 transceiver power dissipation.

KARP, B. AND KUNG, H. 2000. Gpsr: Greedy perimeter stateless routing for wireless networks. In *Proc. of MobiCom 2000*. ACM, N.Y., 243–254.

KRISHNAMACHARI, B., BEJAR, R., AND WICKER, S. 2002. Distributed problem solving and the boundaries of self-configuration in multi-hop wireless networks. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS'02)*. IEEE., Big Island, Hawaii, USA.

US WIRELESS CORPORATION. http://www.uswcorp.com/USWCMainPages/our.htm.

MEGUERDICHIAN, S., KOUSHANFAR, F., POTKONJAK, M., AND SRIVASTAVA, M. B. 2001. Coverage problems in wireless ad hoc sensor networks. In *Proc. of IEEE Infocom 2001*. IEEE, Anchorage, Alaska.

POTTIE, G. J. AND KAISER, W. J. 2000. Wireless integrated network sensors. *Communications of the ACM 43*, 5 (May), 51–58.

PRIYANTHA, N., CHAKRABORTY, A., AND BALAKRISHNAN, H. 2000. The cricket location support system. In *Proc. of ACM MobiCom 2000*. ACM, Boston, MA, 32–43.

RAPPAPORT, T. S. 1996. *Wireless Communications: Principles and Practice*. Prentice Hall PTR.

RF MONOLITITHICS, I. http://www.rfm.com.

ROYER, E. AND TOH, C. 1999. A review of current routing protocols for ad-hoc mobile wireless networks.

SAVVIDES, A., BOULIS, A., KOUSHANAFAR, F., POTKONJAK, M., KARAVAS, V., AND SRIVASTAVA, M. B. 2000. Dynamic location discovery in ad-hoc wireless networks. Tech. Rep. TM-UCLA-NESL-2000-07-001, University of California at Los Angeles. July.

SAVVIDES, A., HAN, C., AND SRIVASTAVA, M. B. 2001. Dynamic fine-grained localization in ad-hoc networks of sensors. In *Proc. of ACM MOBICOM '01*. ACM., Rome, Italy.

SCHURGERS, C., TSIATSIS, V., AND SRIVASTAVA, M. B. 2002. Stem: Topology management for eneregy efficient sensor networks. In *IEEE Aerospace Conference 2002*. Big Sky, MT, USA.

THRUN, S., FOX, D., BURGARD, W., AND DELLAERT, F. 2001. Robust monte carlo localization for mobile robots. *Artificial Intelligence*.

VOR. Very high frequency omnirange. http://www.allstar.fiu.edu/aero/VOR.htm.

WARD, A., JONES, A., AND HOPPER, A. 1997. A new location technique for the active office. *IEEE Personal Communications Magazine 4*, 5 (October), 42–47.

WELCH, G., BISHOP, G., VICCI, L., BRUMBACK, S., KELLER, K., AND COLUCCI, D. 1999. The HiBall tracker: High-performance wide-area tracking for virtual and augmented environments. In *Symposium on Virtual Reality Software and Technology*. 1–10.

XU, Y., HEIDEMANN, J., AND ESTRIN, D. 2001. Geography-informed energy conservation for ad hoc routing. In *Proc. of ACM MOBICOM 2001*. ACM, Rome, Italy.

YE, W., HEIDEMANN, J., AND ESTRIN, D. 2002. An energy efficient mac protocol for wireless sensor networks. In *Proc. of IEEE Infocom 2002*. Vol. 2. IEEE, New York, USA.

YE, W., VAUGHAN, R. T., SUKHATME, G. S., HEIDEMANN, J., ESTRIN, D., AND MATARIC, M. J. 2001. Evaluating control strategies for wireless-networked robots using an integrated robot and network simulation. In *Proc. of ICRA-2001*. Seoul, Korea.

A. BASE LOCALIZATION SYSTEM

In this Section, we describe our base localization system based on beacons. Our client nodes use just one sensing modality for localization, which is radio proximity.

A.1 Localization Methodology

A.1.1 *Idealized Radio Model.* We have found an idealized radio propagation model useful for predicting bounds on the quality of localization. This idealized radio propagation model amakes two rather unrealistic assumptions.

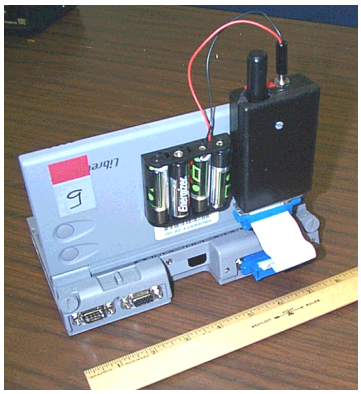
—Perfect spherical radio propagation.

—Identical transmission range (power) for all radios.

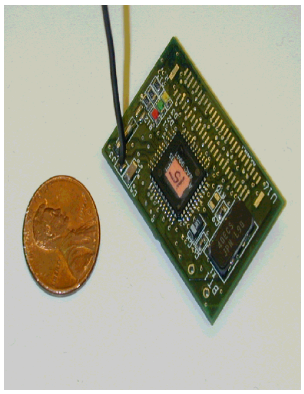
To our surprise, this model compared quite well to outdoor radio propagation in uncluttered environments [Bulusu et al. 2000].

A.1.2 *Connectivity Metric.* In practice, a small fraction of the radio transmissions of a beacon can be received even outside its nominal radio transmission range owing to unpredictable multipath effects. Therefore, it is important to characterize radio connectivity based on statistical observation over a sample of several packets, rather than just one single packet.

A.1.3 *Localization Algorithm.* Beacons situated at known positions, (X_i, Y_i), transmit periodically with a time period T. Clients listen for a period $t >> T$ to evaluate connectivity. If the percentage of messages received from a beacon B_i with range r_i in a time interval t exceeds a threshold $CMthresh$, that beacon is considered connected at r_i.



(1) Librettos with Radiometrix RPC radios



(2) Rene Motes

Fig. 15. The two experimental testbeds used for developing our localization methodology. The experiments described in this paper use the motes testbed.

When the beacon placement is uniform, the centroid of the positions of all connected beacons is a feasible solution in the region of connectivity overlap. A client estimates its position (X_{est}, Y_{est}) to be the centroid of the positions of all connected beacons.

$$w_i = \left(\frac{\left(\frac{1}{{r_i}^2}\right)}{\Sigma_{i=1}^{k} \left(\frac{1}{{r_i}^2}\right)} \right) \tag{18}$$

$$(X_{est}, Y_{est}) = \left(\Sigma_{i=1}^{k} w_i \cdot X_i, \Sigma_{i=1}^{k} w_i \cdot Y_i\right) \tag{19}$$

For non-uniform placement, a feasible solution can be found using more general convex optimization techniques [Doherty et al. 2001].

Given the actual position of the client (X_a, Y_a), we can compute the accuracy of the localization estimate or the *localization error* $LE_{\mathcal{B}}(X_a, Y_a)$, which is the distance between the client's estimated and actual positions.

$$LE_{\mathcal{B}}(X_a, Y_a) = [(X_{est} - X_a)^2 + (Y_{est} - Y_a)^2]^{\frac{1}{2}} \tag{20}$$

A.1.4 *Complexity Analysis.* In our system, both the communication and computation complexity for a device to infer its position once are $O(k)$, where k is the number of beacons in radio range. Because both the computation and communication complexity grow linearly with the density of the beacon infrastructure, rather than the size of the system, our system is generally scalable.

A.2 Implementations

We have implemented a prototype of our localization methodology on two experimental testbeds, shown in Figure 15.

(1) Radiometrix RPC radios connected to laptops via. a serial interface.
(2) UC Berkeley Rene motes [Hill et al. 2000], completely integrated with RFM [RF Monolitithics] radios.

A.2.1 *Radiometrix RPCs.* Our first experimental testbed consists of Radiometrix RPC 418 (radio packet controller) modules connected to a Toshiba Libretto running RedHat Linux 6.0. In our experiments, one of these modules is used as a receiver and four are used as beacons. A 3 inch antenna is used for the experimental purposes. The software for the Radiometrix RPC-418 modules consists of two components.

—*Beacon*: The beacon periodically transmits a packet (every 2 seconds in our experiment) containing its unique ID and position.

—*Client*: The receiver obtains its current measured position based on an input from the user. For each measured position, it samples for a time period t determined by the sample size S, and logs the set of beacons it hears from and its current localization estimate.

A.2.2 *Rene Motes.* We have conducted further experiments on very small, embedded devices called motes, developed at the University of California, Berkeley [Hill et al. 2000]. These devices have a RISC-like 8-bit CPU that runs at 4MHz. Motes are equipped with 512 bytes of SRAM, 256 Kbits of EEPROM, and a 916 MHz ISM radio (RF Monolithics TR1000) that can transmit at the rate of 10Kb/s. The transmit power level of the radio can be controlled using a digital potentiometer on the mote.

Motes can also be programmed as *Beacons* and *Clients*. But because motes have limited storage for experimental data, they can be programmed in two other configurations: Snoopers and Logger. A Snooper mote acts as a network interface for a PC via the RS-232 interface and can listen to all transmitted data packets and forward this to the PC. A Logger mote records all messages sent out by beacons into an EEPROM, and can transfer this information on demand to a Snooper mote connected to a PC.

B. MEASUREMENT TOOLS AND TECHNIQUES

In this Section, we discuss the tools used to measure and evaluate localization quality and our improvements in the localization systems.

While the basic localization software consists of just two components (*beacon* and *client*), the experimental testbed to measure it is actually a lot more elaborate.

We made the following extensions to basic localization software components.

—*Beacon:* It not only transmits periodic advertisements with its position, but it also listens to and responds to commands from a *Beacon Remote Controller*.

—*Client:* It not only estimates its position from the beacon advertisements it receives (the algorithm for doing this position estimation has been described in Section A), but also reports its estimated position to the *Beacon Interpretor*.

We developed the following control and visualization tools.

—*Transceiver:* The *Transceiver* communicates with the Beacons as well as the Clients. It is needed to send control packets to beacons and to interpret the estimated coordinate sent by the Client. To avoid interference, problems with beacons' life-time, etc. the beacons can be remotely controlled. The following commands are supported.

STANDBY	In this mode, a beacon sets its clock rate to the slowest and does not transmit its coordinates. When it receives an ON command, it will enter the NORMAL mode again.
TRANSMISSION POWER CONTROL	It can adjust its own transmission power. (valid in NORMAL mode only)
TRANSMIT RATE CONTROL	It can adjust the rate at which it transmits its coordinates (valid in NORMAL mode only)
RANDOMIZED RESUME	Since a command packet floating on the air can control more than one beacon, each beacon in the STANDBY mode will enter NORMAL mode x milli-seconds after it hears the command, where $x \in (0, y)$ milli-seconds and y is adjustable.

—*Beacon Remote Controller:* A Java GUI that lets the users select the desired TRANSMISSION POWER and TRANSMIT RATE settings that beacons operate at. This program will encode the information and send it as a command packet to the beacons. ON and OFF commands can be selected to toggle the beacons between STANDBY and NORMAL modes.

—*Beacon Interpretor:* A Java GUI that listens for coordinates the Client report and puts them in a table. A user can manually enter the *actual* coordinates of the Client is located and these coordinate will be compared with the *estimated* coordinate to compute the localization error.

—*Visualization:* The visualization program displays all transmitting motes in our laboratory. This is useful to verify whether beacons are transmitting as expected, or if other motes are transmitting and interfering with the experiment.

By allowing us to automatically control and configure the beacons, these tools allow us to experiment rapidly under different beacon density and interference settings without having to reprogram and redeploy all the beacons.