Lessons from Developing and Deploying the Cricket Indoor Location System

Hari Balakrishnan, Roshan Baliga, Dorothy Curtis, Michel Goraczko, Allen Miu,
Bodhi Priyantha, Adam Smith, Ken Steele, Seth Teller, Kevin Wang

MIT Computer Science and Artificial Intelligence Laboratory (CSAIL)
http://nms.lcs.mit.edu/cricket/

November 7, 2003

Abstract

The Cricket indoor location project has been active for four years. We have developed three different versions of the system. The first version was an early proof-of-concept (Cricket v0), which led to the first prototype (Cricket v1). Cricket v1 has seen extensive use by us and by a few other research groups in the community. During this time, we have learned a number of lessons from application designers, users, and system maintainers. We break these lessons into *platform flexibility*, where we discuss the Cricket API, embedded software platform, and hardware interfaces; *location accuracy*, where we discuss Cricket v1's performance and limitations, and *deployment issues*, where we discuss energy consumption and system management. We discuss how these lessons have helped improve the design of the next generation of Cricket, Cricket v2, whose key features we detail. Like Cricket v1, the Cricket v2 hardware design and software will be released as open-source; v2 units will also be commercially available by early 2004. We believe that the lessons described in this paper will be useful to people interested in building or using indoor location systems.

1 Introduction

In Fall 1999, we started work on the design and implementation of the Cricket indoor location system, motivated by the importance of mobile and context-aware applications in pervasive computing environments and the poor indoor performance of the Global Positioning System (GPS). The first version of our system, *Cricket v0*, was a proof-of-concept research prototype [13]. As we started deploying Cricket and obtaining users, we implemented a number of refinements and enhancements, leading to several different subversions of *Cricket v1*. We disseminated Cricket v1 units to a few groups within and outside MIT for research and educational purposes. This process occurred over nearly three years, during which time we learned a number of lessons from the different uses to which the system was put.

This paper documents these lessons, presented as a combination of practical anecdotes and quantitative experimental data. We hope that our experience will be useful to people interested in either using or building indoor location systems in the future. We have organized the lessons into three broad categories:

1. **Platform flexibility.** Cricket was used in several ways that we had not envisioned in the original design. These uses led to our understanding different types of useful location information, a software API encompassing this information, and appropriate hardware interfaces (Section 3).
2. **Location accuracy.** Many applications require accurate location information and do not handle errors well. In particular, many applications do not cope well with high variance in reported location. We discuss our efforts to improve location accuracy and reduce variance (Section 4).
3. **Deployment and management.** Users generally approved of Cricket's decentralized deployment model, which allowed them to quickly deploy a small-scale Cricket system and get going. However, Cricket v1's energy consumption and configuration methods taught us that there was significant room for improvement before long-term production use became feasible (Section 5).

We have attempted to be balanced in our evaluation of Cricket's design decisions, but apologize for any overly defensive comments! At this stage, it would be premature to write a definitive paper on the lessons learned from the project, because it is still ongoing and we don't yet have enough serious users. This paper should therefore be viewed as a status report and as a summary of what Cricket v1 did right and wrong in our estimation.

In Spring 2003, we incorporated these lessons into the design process for the next generation system, *Cricket v2*.

We discuss how Cricket v2's design addresses most of the observed shortcomings of the previous version. The hardware design and software for Cricket v2 will be freely available on the Cricket project's Web site, and units will be available for sale by early 2004.

2 Cricket Overview

The original design of Cricket was motivated by four goals:

1. **Scalability:** Our goal was to scale well to large numbers and high densities of devices requiring location information.
2. **Privacy:** We wanted a system that would make it hard to track users, avoiding the user privacy problem inherent in previous location systems (*e.g.*, Xerox PARC's pioneering Active Badge system [24, 11]).
3. **Low cost:** We wanted to build devices from commercial off-the-shelf components, at a cost of tens, rather than hundreds, of dollars.
4. **Accurate space detection:** At first we were interested only in accurately demarcating boundaries between application-defined spaces (typically rooms or parts of rooms), which is sufficient for many applications (*e.g.*, resource discovery).

Our goals led us to an architecture that was radically different from existing indoor location systems like the Active Badge or Active Bat [6], which use passive ceiling-mounted receivers that obtain information from active transmitters carried by users. The Cricket architecture "inverts" the architecture of the Active Badge and Active Bat systems; in Cricket, ceiling or wall-mounted *active beacons* send periodic chirps on a radio frequency (RF) channel, providing location information to *passive listeners*.

The listener attached to a *host device* (*e.g.*, mobile handheld, portable laptop, sensor, etc.) estimates its distance from each beacon it hears, and uses these distances to infer its location. Each beacon sends an ultrasonic (US) pulse at the same time as the RF message; the listener uses the standard "time difference of arrival" technique by observing the time lag between the arrival of the RF and US signals, to estimate its distance from the beacon.

Qualitatively, the Cricket architecture offers the following advantages:

+ **Good scalability.** The RF and US channel use is independent of the number of listening devices in any region; when host devices actively transmit, high-density deployments are harder to achieve.

+ **Ease of deployment.** Cricket beacons are easy to deploy; they do not require any infrastructure connecting them back to a base station, and can be placed with few constraints inside rooms, open areas and corridors.

+ **User privacy.** Cricket's architecture allows a host device to infer its location without the infrastructure or any other entity learning that information. While Cricket by itself cannot guarantee user privacy, it makes centralized tracking of users hard.

These advantages come at some cost:

– **Continuous tracking is harder.** In Cricket, a listener hears only one beacon at a time. Updating the position of a moving device is more complex than in a system that simultaneously obtains multiple distance estimates from the device to known positions.

– **Beacon scheduling requires a distributed scheme.** Cricket requires a distributed beacon scheduling scheme to avoid RF and US collisions at the listeners.

– **Energy consumption is potentially higher.** Active beacons tend to consume more energy than passive ceiling-mounted receivers. However, both architectures require the transmitters or receivers distributed in the infrastructure to be powered somehow.

2.1 Project Timeline and Current Status

After some experience with Cricket v0 units from the fall of 1999 through the spring of 2000, we started the design of Cricket v1 (4 MHz Atmel processor and a 418 MHz AM-based Lynx radio) in the summer of 2000. We had working hardware by the fall of that year, and a small number of other users within a multi-group collaborative effort at MIT by the spring of 2001. By this time, we had also embarked on the design of the Cricket compass to provide orientation capabilities. Between Fall 2000 and Spring 2003, we made several small changes to Cricket v1, and produced several hundred beacons and listeners for use by a number of groups, including ourselves. In early 2003, we started designing Cricket v2 (8 MHz Atmel processor and a Chipcon CC1000 radio). We have two different v2 hardware designs, one produced by us and one produced in collaboration with Crossbow Technology. Figure 1 shows Cricket v1, the two kinds of v2 boards, and a compass board that attaches to a listener.

The start of our effort on Cricket coincided with a large, lab-wide research effort in pervasive computing at MIT and Cricket soon became a core technology in that effort. Versions of Cricket have been used by several groups at MIT for applications including people location, multi-player physical/virtual games, human and robot navigation, stream migration, and also for several student projects in an undergraduate pervasive computing

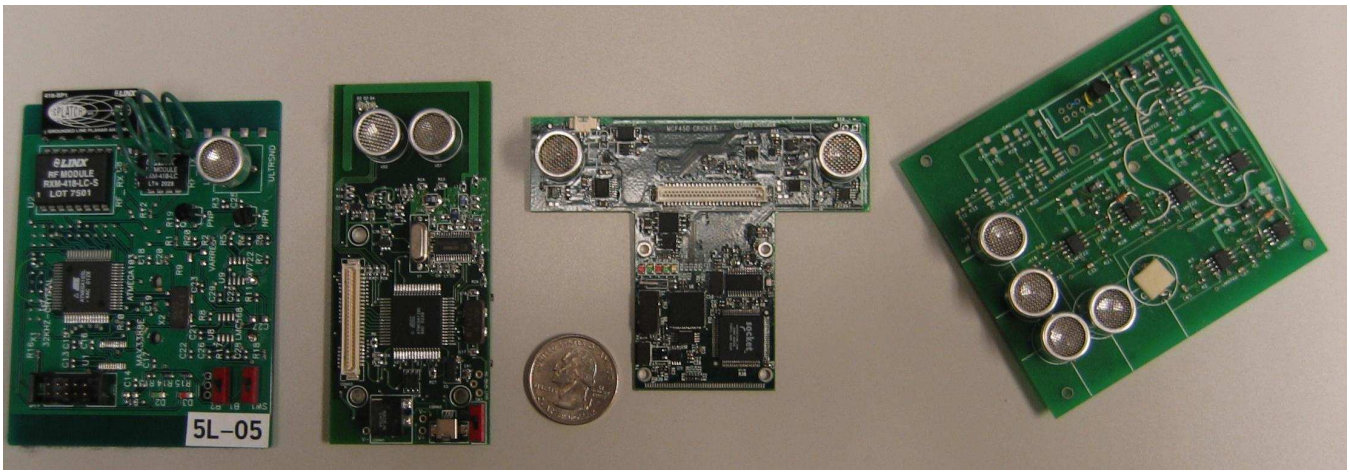

Figure 1: **A few different Cricket units. From left to right: v1, v2, v2 done jointly with Crossbow, and a compass daughter board. The T-shape of the new v2 units enables them to fit into a compact flash (CF) slot on a handheld or laptop. The v1 and v2 units can function as either beacons or listeners.**

course. We have also offered Cricket courses, held at MIT and in elsewhere.

In addition to groups at MIT we have distributed Cricket units to researchers elsewhere, including NTT Labs, Nokia Research, Delta Electronics, the Acer group, Crossbow Technology, Rutgers University, University of Washington, Intel Research, Philips Research, and HP Labs (in addition to using our hardware, the last two have also made their own versions with different radios). We have had over a hundred requests from other potential users for Cricket devices, requests that we have been unable to satisfy with our limited production capability.

Our experiences with users and applications has informed our design of Cricket v2. For example, some applications require spatial information, some require fine-grained, GPS-style position coordinates, and some benefit from orientation in addition to position. In addition, location-aware applications are not restricted to mobile handhelds or laptops—many embedded sensor network systems and applications require location information as well.

2.2 Some Cricket Applications

This subsection describes some of the applications that people have built using Cricket. This is not an exhaustive list, but is intended to convey the diversity of applications and point out limitations in previous (sub)versions of Cricket.

2.2.1 Applications using "space" ID alone

The first class of applications uses only information about the space (*e.g.*, the room, or part of a room) that the mobile device is currently in. These applications rely on Cricket's ability to demarcate physical and virtual boundaries between spaces. Each beacon periodically broadcasts its space name on Cricket's RF channel. Any listener device reports the nearest beacon it hears. We found the space abstraction useful in several applications.

Resource discovery. Our first Cricket application was location-aware resource discovery, which we prototyped in conjunction with a separate resource discovery system. The goal is to find resources based on attribute-value queries, and have The discover system perform the matching between queries and resource advertisements. Location, obtained using Cricket, is an important attribute, because users often care about obtaining access to resources near where they are.

Pervasive Access Control. A student at MIT developed a pervasive access control system, *PAC*, using the coarse-grained location information provided by Cricket. This system provides light-weight access control based on location, while preserving the user's anonymity. For example, this infrastructure allows a service to be built so that the ability to remotely control a room's projection or lighting system would be honored only for users who could prove that they are currently close to that resource. To implement this secure location subsystem, he assigned each beacon a location id (LID) and a LIDCODE, a pseudo-random number based on a seed. The LIDCODE changes every minute in pseudo-random fashion, and the beacon periodically transmits its LID and LIDCODE. A device can present the appropriate LIDCODE to a server only if it is near the concerned space (or if someone near the space gives it the code).

Person locator. The person locator has two components: an application running on a handheld device equipped with a Cricket listener and a wireless network card, carried by each person, and a central server that can

be queried via a Web or conversational speech interface. When the application on the handheld device detects that the user has moved to a new location (*i.e.*, when the identity of the closest beacon changes), it securely updates the user's location on the server. The user can control when, and if, their handheld reports their location. The user can also set preferences on the server to determine the level of detail reported based on their location and the identity of the person making the query.

Stream migration. Three different stream migration services have been built with Cricket. The first, migrates a live video conference to the best available display/sound resource.

The application uses Cricket to detect transitions into rooms where video conferencing resources are available, at which time the video and audio streams are migrated from the handheld device to a more suitable device in the room. When the user leaves the room, the video conference is migrated back to the handheld. The application developers found that simply using the nearest beacon to determine room identity was not sufficiently stable. One noisy sample could make a beacon in the hallway appear closest when the user was in fact in the room. This would cause video to start migrating to the handheld, then immediately back to the room. Two other stream migration systems implemented with Cricket faced the same location stability issues: audio stream migration [14] and live television migration [15].

It turned out that these and other application writers wanted access to the information provided by the beacons heard by the listener, in order to implement application-specific filtering and hysteresis. With this information, they were able to make application-specific tradeoffs between stability and responsiveness. As expected, the more samples used for averaging the more stable the system is, but it then takes longer to recognize that a new room has been entered. In Section 3, we discuss how this experience was reflected in the Cricket API.

2.2.2 Applications requiring position coordinates

A second class of applications uses beacons that broadcast pre-programmed fine-grained (2D or 3D) position coordinates. Here, a listener hearing sufficiently many beacons[1] can solve for its own location.

CricketNav is a mobile indoor navigation application that runs on wireless handheld computing devices [16]. CricketNav uses Cricket to track the user's position in real-time and help users navigate by displaying a sequence of arrows leading to the desired destination. People can use CricketNav to locate a particular place, person, or resource in an unfamiliar or complex environment.

[1]Three or four, depending on the 2D or 3D nature of the application and whether the speed of sound in the local environment is accurately known.

Our original goal was for the navigation application to use *only* space information; we reasoned that human users apprehend space and room information more readily than position coordinates. So, to help a user navigate, the application would display a list of spaces to traverse. However, we quickly discovered that augmenting spatial information with position coordinates improved CricketNav's usefulness. For example, it was often useful to know exactly how far away from a door within a room a user was, or how close to a turn they were. As a result, we added position estimation capabilities to Cricket listeners, based on information about known beacon coordinates.

CricketNav uses spatial information as a convenient handle to fetch the relevant maps from a map server. The spatial information also helps CricketNav give better directions. When the user moves near wall boundaries, it is often difficult, using coordinate information alone, for applications to determine which side of the wall the user is on. This is because coordinate information is often associated with a margin of error that can overlap a wall boundary. Consequently, the navigation system may determine that the user is on the wrong side of a wall and generate incorrect directions. Because the spatial estimate does not suffer from the same error modality, the combination of space and position information usually correctly disambiguates the user's postition.

CricketNav made it apparent that Cricket v1's high variance in position estimation sometimes made the application sluggish or unusable. Furthermore, Cricket sometimes did not provide location with enough accuracy, and was unable to determine when it was giving inaccurate information. This experience motivated our successful efforts to improve the accuracy, and reduce the variance, of Cricket v2 by more than tenfold (see Section 4).

Physical computer games. We also found that users want to build applications that require a moving device to accurately track its position while in motion. As part of a pervasive computing course [17], the instructor and his students (who were not involved in the Cricket project) used the Cricket infrastructure to develop a combined physical/virtual version of the popular computer game, "Doom" (Figure 2). This application used Cricket to track a player's movement within a room and to reflect that movement into movement in the game.

In Section 3, we discuss how this experience led to changes to allow users to modify the Cricket firmware in more convenient ways than originally supported. In Section 4, we discuss approaches to improve Cricket v2's tracking performance, including the different options used for the game application.

2.2.3 Pose-aware applications

Cricket devices and listeners can be configured to provide fine-grained "pose" information, defined as combined po-

Figure 2: **Screen shot of a Cricket-enabled Doom game developed in a pervasive computing course at MIT.**

Figure 3: **A prototype "software marker" (a software compass integrated with a laser range-finder).**

sition and bearing. A variety of prototype "pose-aware" applications have been developed within the Computer Graphics Group at MIT [18]. These applications led us to develop the Cricket compass. The compass infers a device's orientation by using multiple US sensors to obtain differential distances to one or more beacons [19].

The pose-aware applications described below do not yet use the integrated Cricket compass, because we have not yet managed to mass-produce compass units (we have only made bench prototypes at this time, one of which is shown in Figure 1). In Section 4.4 we describe the problems with the original Compass design (which worked, but was hard to manufacture in bulk) and how our new design will improve manufacturability.[2]

[2]We expect to disseminate compass units as attachable boards to Cricket v2 late in 2004.

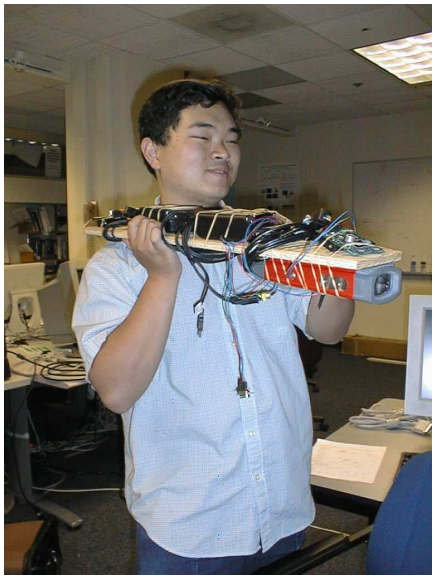

Figure 4: **A shoulder-mounted "software marker."**

Currently, these applications use a "hand-held" (actually, shoulder-held, and colloquially called the "Cricket bazooka") prototype compass, made from two position listeners separated by a fixed one-meter baseline (Figure 4). A PDA computes the compass's midpoint and bearing simply by computing the average and difference, respectively, of the position listener's reported coordinates. (When integrated Cricket compass units are available in bulk, these applications will migrate to that platform.)

Improved navigation. The most immediate use of pose-awareness is to provide improved navigation services, in which the user's handheld device can show the user's position and desired direction of motion in context (much as existing heads-up navigation displays do in high-end cars).

Software marker. In concert with a geometric environment model, a pose-aware device enables the user to indicate a structural element (portion of wall, floor, ceiling, or doorway) of the environment simply by pointing at it. The application can then provide query or annotation capability based on the inferred (2D or 3D) location of the indicated element (where the inference is made by casting a ray from the device's location, in the reported direction, until the ray encounters a modeled surface element).

With the addition of a hand-held laser range-finder (left portion of Figure 4), the application can infer the (2D or 3D) location even of unmodeled elements, for example moveable furniture or computers. The application can then associate the object's current position with metadata (*e.g.*, ownership information) in a spatial or relational database.

Software flashlight, for direct information overlay. With the addition of a digital projector, a pose-aware ap-

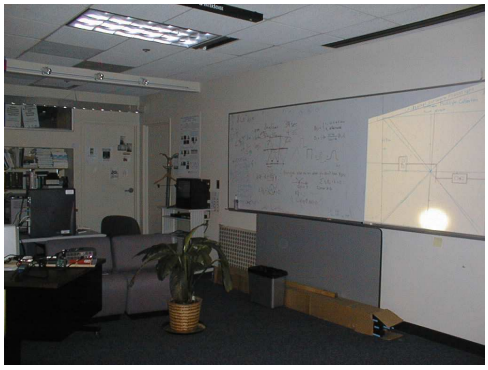

Figure 5: **Direct information overlay: a pose-aware projector (left) overlays geometric information (planned electrical outlets) onto an existing wall (right).**

plication can perform "direct information overlay" by projecting textual or geometric metadata directly onto environmental surfaces (see Figure 5). The information could be textual (*e.g.*, a maintenance history) or geometric (*e.g.*, installation or repair diagrams).

We envision using the direct overlay device as a handheld tool, to be carried on a tool belt and used intermittently as needed. Like a flashlight, the tool could be either hand-held, or rested on a surface such as a table-top for hands-free operation, for example to illuminate a work area. A typical usage scenario would be for routine maintenance: a user notices a problem (for example, a malfunctioning power outlet), and indicates its location using a software marker. The spatially coded maintenance request enables the maintenance person to navigate to the trouble spot, using the software compass. Finally, the maintainer uses the software flashlight to illuminate the problem area, and show the routing of wiring within the wall, and its path to the nearest breaker box.

2.3 Location-aware Sensornet Applications

We have also found that many embedded sensor network applications and protocols can benefit from location-awareness. Access to location information is useful in routing, data dissemination, sensor stream annotation, etc. We have concluded that it is important for an indoor location system to work with both handheld mobile computing devices and sensor computing nodes. In Section 3, we discuss how Cricket v2's design accommodates both possibilities.

The space-based, position-based, and pose-aware applications described in this section taught us several things about Cricket. We break the different lessons into platform flexibility issues (*e.g.*, API issues, access to firmware, physical connector issues, etc.), location accuracy and performance issues (*e.g.*, improving steady-state accuracy, better outlier rejection for reducing variance, better tracking performance, and a more robust compass), and deployment issues (*e.g.*, energy consumption and system configuration). We discuss these issues in the next three sections.

3 Platform Flexibility

As discussed in the previous section, we found several other user needs beyond our original plan:

1. Providing position coordinates.
2. Providing orientation.
3. Enabling application-specific filtering and hysteresis on location data.
4. Providing reasonable performance for continually moving users.
5. Providing "power users" access to the firmware to change things like the beacon scheduling method.
6. Providing location information to sensor nodes.

To handle the first requirement, we enhanced Cricket beacons to disseminate their position coordinates in addition to space (actually, as we explain in Section 5, Cricket beacons don't disseminate their coordinates; we found it much more convenient to have applications query a beacon ID database that maintains mappings between beacon ID and beacon coordinates). [3]

The previous section also described how we handled orientation needs (albeit in a somewhat clumsy way, pending the development of a more robust compass).

3.1 Software API

"Raw" access. Our first few users told us that the original idea of having the Cricket listener perform all the filtering of beacon information and provide only the closest beacon to the application was not a good idea. We modified Cricket v1 to provide a simple and general API: the listener passes all distance samples from each beacon to the attached host device. The host device (either some "middleware" or the application itself) implements all the processing to infer the host's location. We found this to be a good design decision, because different applications processed raw distance samples in different ways, even when they were all interested in space information.

Cricket v2 continues to provide raw access to the information collected at the listener to host applications. Additionally, Cricket v2 listeners will also perform a significant amount of embedded processing, including implementing a Kalman filter for tracking moving nodes. This processing will allow v2 listeners to be used with a variety of host devices including sensors that don't perform any Cricket processing.

Information fidelity. A deployed Cricket infrastructure, like GPS, does not always provide perfect location and orientation information. Rather, the fidelity of location information may degrade under a variety of circum-

[3] If implemented carelessly, this could comprimise privacy. In particular, the database should be downloaded in full, rather than be queried.

stances. For example, hearing only RF message without any accompanying ultrasound would place the device in a range because there would be no distance estimates, but it is still useful information to applications. Or, depending on device movement and ambient ultrasonic noise or reflections, the listener may have reduced confidence in the accuracy of its distance estimates. As another example, ultrasound noise reduces the orientation listener's ability to discriminate arrival phase at multiple ultrasound receivers, reducing the accuracy of the listener's orientation estimate. If the position listener hears an insufficient numbers of beacons, it will be unable to trilaterate to determine its own position. In this case, the listener can still report coarse-grained location estimates by reporting the identity of the closest single beacon.

These circumstances form a hierarchy of fidelity levels, which an application can use both to adjust its operation, and to inform the user so that s/he can adjust expectations appropriately. These fidelity levels include fine- and coarse-grained pose, fine- and coarse-grained position (no orientation), stale pose (accurate information, but the device has moved since its most recent report), and an out-of-service area where the listener is far from any beacons.

We are developing ways to bridge short-term service dropouts by integrating one or more additional sensors to the hand-held listener. A tilt sensor, gyro, or accelerometer can provide relative attitude or location information for a few seconds. An outward-looking camera can track the device's "egomotion" or rigid-body motion indefinitely (up to a single, unknown translational scaling factor), provided that the environment contains sufficient texture or geometric information. Finally, if the device can identify and track known features in the environment (edges, corners, door-frames), it can solve for its pose independently.

Reporting age. We found it extremely useful for the listener to report *age* information for every beacon, because beacon broadcast collisions, beacon scheduling, and packet losses introduce significant latency between chirps. Applications can use this information in different ways, *e.g.*, to interpolate or extrapolate the device's current location based on how users are likely to move while running any given application. For example, while playing a game in front of a large display, it is unlikely, although not impossible, for the user to go into a different room altogether. In the person locator application, age information was used as input to the hysteresis to determine when someone had left a room.

3.2 Software platform flexibility

In Cricket v1, we had erroneously assumed that users would not be interested in changing the firmware running in the beacon and the listener. We found, however, that some users wanted to make changes to beacon scheduling, listener filtering, etc. The use of a commercial compiler, and software that was tightly coupled to the underlying hardware, made such changes both expensive and time consuming. To overcome this shortcoming, we have rearchitected Cricket v2's embedded software and implemented it in the TinyOS environment [23]. In addition to easier development, the move to TinyOS is likely to make it easier to develop location-aware sensor network applications using Cricket.

This change required signigicant effort: Cricket precision depends on the accuracy of mesurement of the time interval between the RF and the ultrasound arrival times. The TinyOS event driven architecture is not well-suited for such precise timing of events. Achieving the timing granularity of Cricket v1 with TinyOS required implementating Cricket's wireless messaging deep in the TinyOS radio code. It also required the addition of a capture pin on the embedded microprocessor for microsecond timing.

3.3 Hardware interface

Cricket v1 listeners interface to a host using a RS232-serial interface. This turned out to be inconvenient for mobile users because it required an unwieldy and obtrusive cable, and was a barrier to wider adoption. Cricket v2 provides a more convenient compact flash interface. The compact flash provides a solid attachment to the host. It also provide power to the v2 listener, eliminating the need for a battery pack. To enable easy integration with sensor platforms, Cricket v2 also provides a connector to the Berkeley mote / Crossbow Mica platform.

This design (see Figure 1) also opens up the possibility of *mobile sensors*, where a handheld computer with a Cricket listener in its CF slot, to which a commercial sensor board is attached, can be carried by users and also act as sensors in addition to being used for human-centric mobile applications.

4 Location Accuracy

In Section 2.2, we described a few shortcomings of Cricket v1 in terms of its accuracy and precision. Cricket v2 fixes several shortcomings of v1 based on our experience with several applications. First, because Cricket v1 was primarily optimized for good spatial boundary detection, its position accuracy in real deployments had high variance, being accurate to only about 30-40 cm. Cricket v2 improves this significantly, being able to obtain distance estimates to within 1 cm on average and 3 cm most of the time (see Figure 6).

4.1 Improving distance estimation accuracy

The Cricket v1 listener used a "phase lock loop"(PLL) ultrasonic detector (Figure 7(b)). This detector had highly variable detection characteristics, leading to distance mea-

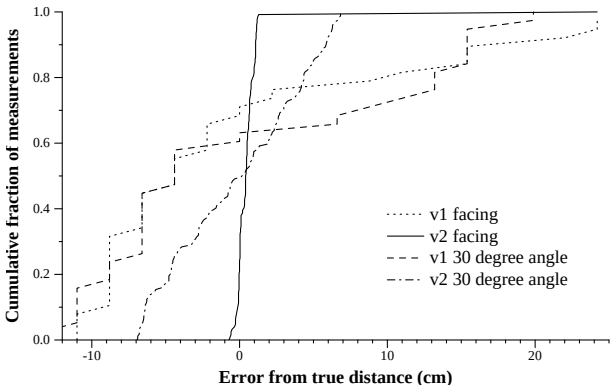

Figure 6: **CDFs of measured distances in Cricket v1 and Cricket v2, showing v2's much-improved accuracy.**

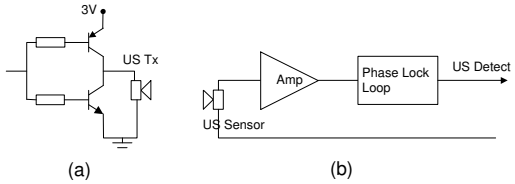

Figure 7: **Cricket v1 US circuitry: (a) beacon and (b) listener.**

surement errors as high as 30 cm. In Cricket v2 we replaced the PLL-based detector with a simpler amplitude detector (Figure 8(b)). This improved the detection accuracy substantially, to about 1 cm, but also reduced the sensitivity of the detector circuit. To compensate for this, we increased the ultrasonic transmitter signal strength by increasing the drive voltage from 3V to 12V (see Figures 7(a) and 8(a)).

The CC1000 radio chip used on the Cricket v2 also presented difficulties because the binary data does not arrive deterministically (the data does not arrive eight bits at a time). The offset of bits arriving late for a given start symbol (representing the start of ultrasound) can change the precision of Cricket by 7 cm. We compensated for this by recording the bit offset of the first byte of the start symbol within the byte captured by the radio and then adjusting the timing in the listener firmware.

4.2 Rejecting outliers

We found that precise distance measurements require sensitive US sensors, but such sensors react to ambient ultrasonic noise and high-energy sound pulses. In particular, we found that malfunctioning fluorescent lights, people jangling keys, and loud noises (*e.g.*, slamming doors) cause the listener to record bad distance samples. Accurate distance estimation therefore requires good outlier rejection methods.

Initially, we used Cricket to determine the location of mostly static objects for resource discovery applications (we assumed a person to be static for several seconds before computing the current location). The MinMode al-

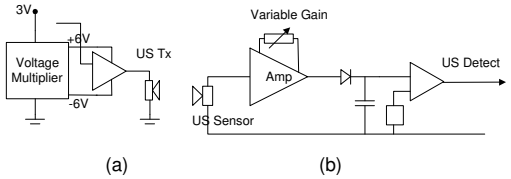

Figure 8: **Cricket v2 US circuitry: (a) beacon and (b) listener.**

gorithm implemented in Cricket v1 has good outlier rejection properties when the listener is static. It first collects distance samples for a fixed time window of 5 seconds, then it rounds of the samples to the nearest 20 cm. Next, it selects the value that has the maximum number of occurences as the true distance; if there are several values with the maximum occurence, it selects the minimum value as the true distance. Although this algorithm performs well when the listener is static, its performance degrades when the listener is mobile because the dynamic distance values prevent the algorithm from obtaining the correct value with a high enough frequency.

In Cricket v2, the extended Kalman filter (explained in the next section) used for obtaining position information while a device is moving maintains an estimate of the variance of the filter's position state. We use this to reject outliers; the variance of the position estimate defines a threshold, and if a sample falls outside of that threshold, the listener rejects it. This approach usually rejects all reflections and noise, unless the state estimate is itself bad. In that case, as explained below, the Kalman filter's state resets, and the "outlying" sample is not rejected.

4.3 Fast tracking of moving objects

The applications described in Section 2.2.2 require fast updates of a moving object's position coordinates. Because Cricket is an active beacon architecture, meeting this requirement is more involved than if it were an active mobile system like the Active Bat. The reason for this is that the *simultaneity condition*— the availability of multiple distance estimates to known position beacon/sensor positions in the infrastructure for the same current position of the moving device—is not satisfied in general.

4.3.1 The "Doom" approach

The initial attempts by the developers of the Cricket-enabled Doom to use multiple Cricket v1 beacons on the ceiling and the full 3D location tracking code did not give a fast enough update rate for game play. New values from at least three beacons were required to calculate each location. The 3D solver did not always give valid results, causing position calculations to be dropped, further reducing the update rate. Also, setting up the game required configuring the position of each of the beacons.

They addressed both issues by simplifying the problem, taking advantage of the way user's would move while playing the game. First, they used only two beacons, one

on each side of the projection screen showing the virtual world, at waist height (blackboard chalk rails where convenient holders). The beacon's code was modified to transmit more frequently, as they where only competing with each other for transmission time. The moving device's location was calculated only in a 2D horizontal plane using the intersection of the two circles centered at each beacon. There is only one valid solution, the second intersection point is always behind the display. Here, responsiveness is more important than absolute accuracy.

The developers of this application also came up with a simple and elegant application-specific beacon configuration method to avoid manual configuration. At initialization, a listener is held very close to one of the two beacons, and measures the distance to the other beacon. The result gives the distance between the two beacons, which is the only parameter needed to configure the system. (In Section 5.2 we show how a more general method solves a more general configuration problem.)

Another way to use Cricket for this application would be to make the moving device an active transmitter. The game developers attempted this method, but given the time constraints of a term project and the added complexity of merging two streams of location updates, they were not able to get the information gathered at two different listeners coordinated at a single location.

This experience, as well as our own experience with CricketNav, suggested a number of improvement possibilities. The rest of this section discusses some of them.

4.3.2 Using an extended Kalman filter

A Cricket v1 listener computes its position by storing the last T seconds of distinct beacon samples and running a least-squares minimization (LSQ) to minimize the residual error. Specifically, if the known beacon position of beacon i is b_i and a distance estimate from it is d_i, the listener estimates its position p by minimizing $\sum_{i=1}^{N}(\|b_i - p\| - d_i)^2$, where $\|b_i - p\|$ is the Euclidean distance between the coordinates b_i and p.[4]

Of course, if the device is in motion, a large value of T leads to inaccuracies because LSQ will use old distances that may not be close to the current position. Moreover, in general, LSQ alone does not adequately capture motion state. GPS faces a similar problem, and handles it using a Kalman filter [5]. Cricket v2 also implements a Kalman filter to help a moving device track its position.

Like GPS, Cricket v2's Kalman filter maintains a state vector that estimates the device's current position and velocity (currently, we don't use higher order terms like acceleration). Unlike GPS, Cricket v2's filter has to operate "single-constraint-at-a-time", because the simultaneity condition does not hold. The idea in the Kalman filter is simple: maintain the device's position and velocity estimates, and assume that between beacon chirps, the device moves at constant velocity. Each beacon chirp defines a time-step. The filter uses a predictor to produce an estimate of the device's position at any time between chirps, and a corrector to rectify the position and velocity state whenever a beacon chirp is heard and the reported distance deviates from what the predictor suggests.

A covariance matrix reflects the filter's confidence in the state vector. With each incoming measurement we update our state vector based on the new data, weighing the state against the new information by comparing the current covariance against the variance of the new measurement.

Once in a while, the Kalman filter's state becomes bad (the covariance matrix has large values), suggesting that its predictions are wrong. After substantial experimentation, we found that a good way to reset the bad state of the Kalman filter is to obtain a new fix (a more accurate position estimate) on the device's position using LSQ on the past small number of beacon samples, ignoring the Kalman filter.

We found that one way to obtain a good fix is to move from a purely active beacon system to a hybrid system where a moving listener would become an active transmitter whenever the Kalman filter's state went bad. To do this, while maintaining Cricket's scaling and privacy goals, requires a new protocol. Cricket v2 uses the following method:

1. In the common case, the listener does not transmit any information, only beacons do.
2. If the listener's Kalman filter state is bad (covariances above a configurable threshold), then it becomes an active transmitter. This usually happens if the device experiences sudden linear acceleration or turn. It generates concurrent RF and US pulse, with the RF message having no information in it other than a randomly generated nonce. This message is sent in a specific timeslot when all the beacons listen on the channel.
3. If a beacon hears an RF message and the corresponding US pulse in the "beacon listening" timeslot, it waits for a short period of time and broadcasts the nonce (set by the listener) together with the distance estimate. The listener hears this information from all the beacons, obtaining good information about its current position because the simultaneity condition now holds.

It is important to note that the transition to an active transmission from the moving listener happens only when the Kalman filter's state is bad, which means that the system as a whole is likely to remain scalable because it is un-

[4] LSQ posed in this manner is computationally very intensive, so the listener linearlizes these equations and approximates the solution. That approximation does not always minimize the true least-squared error, but is usually good enough.

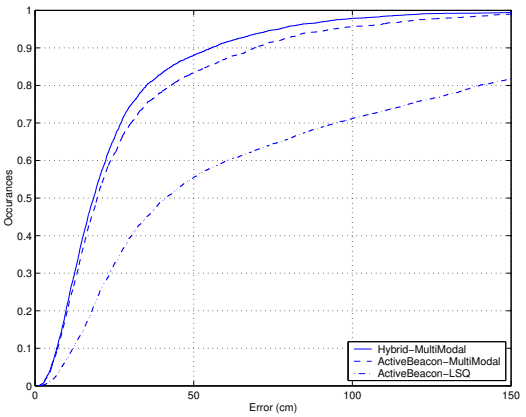

Figure 9: **CDF of tracking accuracy at a speed of about 0.7 m/s for three Cricket schemes; the best scheme is a multi-modal Kalman filter (not described in this paper) that uses the hybrid approach of occasionally going into active-transmit mode. The median error is comparable to an always actively transmitting mobile scheme.**

likely for every listener in a room to simultaneously have a bad filter state. The use of a random nonce does not directly reveal the mobile's identity, and the beacons broadcasting the distance estimates back to the listener solves the problem of correlating these samples.

4.3.3 Conducting experiments

We found it difficult to design a testbed facilitating repeatable experiments. The characteristics of RF and ultrasound depend on ambient conditions, walls, people in the vicinity of the transmissions, etc. All our distance estimation experiments were conducted in a variety of different rooms, lab space, and corridors.

Motion tracking proved particularly problematic, because we wanted to compare the performance of different methods under identical conditions. To do this, we bought a computer-controlled Lego train set and tens of meters of train tracks, and set it up in a large room. We attached a Cricket listener to the train and beacons to the ceiling. We wrote utilities to precisely control the movement pattern and speed of the train, including pause times and random velocities between pauses. We experimented with this apparatus at speeds of up to about 1 m/s. This experimental setup included a number of real-world effects, including multiple beacons (up to six) interacting with one another, varying distances from the different beacons to the listener, and ultrasonic noise and reflections (in fact, we found that the engine of the train generated some ultrasonic noise that the listener had to filter out!).

Figure 9 shows a sample CDF of the tracking accuracy at an average movement speed of 0.7 m/s. The median inaccuracy for the best scheme (the hybrid scheme) is under 20cm, and the lag is one-sided, which means that an application may be able to account for it. Overall, we are

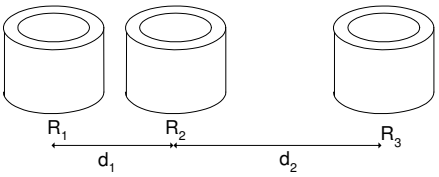

Figure 10: **Ultrasonic sensor array used by the Cricket compass.**

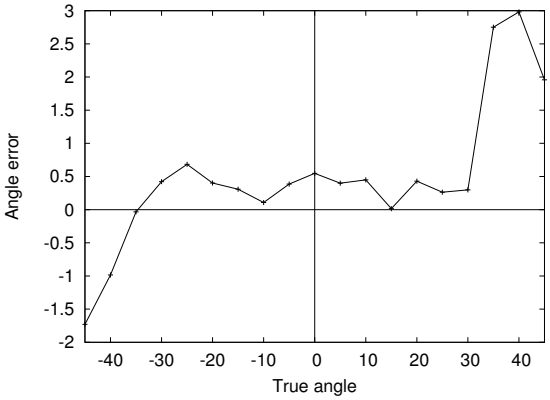

Figure 11: **Cricket compass error vs. true angle.**

happy with the performance of this system, which shows that over a 5-10 meter beacon range, we can track reasonably fast human speeds with acceptably low error. A more complete description of these experiments and Cricket's tracking methods is in [20].

4.4 Improving compass accuracy

We designed the Cricket compass to obtain direction information within the Cricket system. The Cricket compass consists of two arrays of US sensors, each array having three collinear sensors as shown in Figure 10. When a US signal is received, we measure the pair of phase differences (θ_1, θ_2) between the sensor pairs (R_1, R_2) and (R_2, R_3). By selecting proper values for the seperations d_1 and d_2, we can get unique values for (θ_1, θ_2) when the array is rotated by $180°$. These results heavily depend on the accurate placement of the sensors in the aray; if the seperations between sensors change by even 1 mm, the reported angle may have an error of tens of degrees. Although we managed to place the sensors accurately for our prototype, it was not possible for us to place the sensors so accurately when building several hundred units. One solution to this is to build the sensor array during the manufacturing process of the sensor itself, but sensor manufacturers were reluctant to build such custom sensor units for quantities less than tens of thousands of units.

With the increased accuracy of Cricket v2, we changed the architecture of the compass slightly to make it amenable for mass production. We still use the phase difference θ_1 between the receiver pair (R_1, R_2). When the receivers are placed λ (the wavelength of ultrasound)

apart, we obtain the same phase difference for two angles that are seperated by 90°. Since the improved hardware design enables us to measure the distance difference from the beacon to the two receivers R_1 and R_2 with an accuracy of $\simeq$ 1cm, we can use this measured distance difference to differentiate between the two angles that correspond to a given phase difference θ_1. In this scheme, since, all three sensors need not be accurately colliner. Any placement errors when placing R_1 and R_2 can be corrected using a simple calibration step. Figure 11 show the performance of the new compass architecture (we show only -45° to $+45^\circ$ since we use two perpendicular sensor arrays for angle measurement).

5 Deployment and Management

One can easily put together a Cricket location system by attaching a small number of beacons on the ceiling and by connecting a listener to a host running Cricket application software. The ability to rapidly deploy a working location infrastructure has been a significant user-perceived advantage of Cricket. Demonstrations of the system have been relatively easy to perform both on-site and off-site, students have found it easy to make fast progress in class projects, and users have told us that they have found it painless to get going with the system.

There are two main issues in deploying large numbers of Cricket beacons in a production system: power management and configuration management.

5.1 Power Management

Cricket s receive their power from batteries. This is convenient for initial deployment and system demonstrations. However, in Cricket v1, at current power consumption rates, the batteries need to be replaced every three weeks. This is especially inconvenient and potentially costly in manpower for production use in even moderate-sized deployments.

As the size of a Cricket deployment grows, it is important to manage energy consumption better than what Cricket v1 originally did. To this end we considered several approaches: hardware improvements, scheduling optimizations, alternate power sources, as well as the design of a monitoring infrastructure to detect "missing" beacons whose batteries have failed.

5.1.1 Hardware Improvements

Cricket v1 hardware design was not optimized for low power consumption. Since the frequent battery changes became a big hurdle against a permanent deployment of a Cricket network, we designed Cricket v2 to be more power-efficient. Assuming a beacon frequency of 1Hz per beacon, Table 1 shows the current consumption and the operating time of various beacon subsystems for Cricket v1 and v2.

Although Cricket v1 works well at moderate beacon densities, high deployment densities (twelve or more beacons all within range of each other) causes problems. This problem, became apparent in situations where users deployed a large number of redundant beacons to protect against batteries running out. The poor noise immunity of the Cricket v1 radio (amplitude modulation and surface-acoustic-wave based receivers) caused errors in received radio messages, which caused them to be dropped. Cricket v2 overcomes the noise problems using a better radio based on frequency modulation and a superheterodyne receiver (CC1000 from Chipcon), and appears to perform well at high densities.

Cricket v1 used separate RF transmit and receive circuits. We found that when batteries on beacons ran down and the voltage dropped, the receiver unit failed before the transmit unit, causing the carrier sense mechanism to fail and leading to poor performance. We have corrected this problem in v2.

5.1.2 Scheduling Optimizations

We also improved the scheduling of the beacons. In Cricket v1 the chirp schedule was defined by the sleep time of the beacon between attempts to broadcast location information and some random delay.

To prevent collisions between beacons the radio signal lasted from the beginning to the end of the US pulse. A beacon would send its chirp when the radio was free, using carrier sense as a trigger.

In Cricket v2 the radio signal does not envelop the US signal anymore. Instead we have a start of chirp message (SYN1) and an optional stop of message (SYN2). The US pulse is also shorter (150 μs instead of 500 μs) but its lifetime is longer (50 ms instead of 40 ms). This lets us save on energy and provides the possibility for inter-beacon RF communication even when the ultrasound is in flight.

In Cricket v2, the beacons now have a mean chirp frequency and listen for SYN1 messages before sending, instead of using carrier sense. If a SYN1 message is received during the wait period before the chirp, the timer is reset and the beacons wait for the ultrasound life to expire. An extra random delay is added to prevent collisions between chirps when two or more beacons compete for the same chirp time. The wait time is subtracted from the next sleep time to preserve fairness and the average chirp frequency. The state diagram of the chirping process is shown in Figure 12.

To detect interference caused by hidden terminals, the listener discards location measurements if two or more SYN1 messages arrive (from different beacons) inside the lifetime a the ultrasound pulse.

Another approach to scheduling could be to use ultrasound modulation. This would let us send ultrasound

Sub system	Processor	RF Tranmitter	RF Receiver	US Transmitter
Cricket v1 (active) Cricket v1 (idle)	3.7mA×50ms 1.5mA×1000ms	1.5mA×50ms 1.7μA×1000ms	6mA×1ms 0.7mA×1000ms	2mA×250μs 1.5mA×1000ms (idle)
Cricket v2 (active) Cricket v2 (idle)	7mA×45ms 10μA×1000ms	7mA×5.3ms 0.1μA×1000	7.4mA×32ms -	50mA×120μs 0.5μA×1000ms

Table 1: Power consumption of Cricket v1 and v2 beacon subsystems.

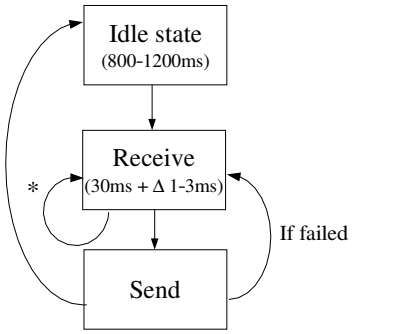

Figure 12: **State diagram of the beaconing process.**

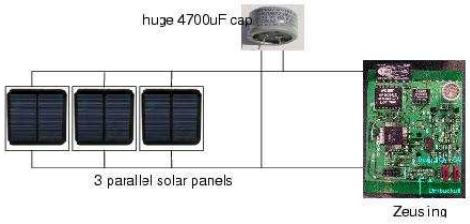

Figure 13: **Cricket v1 with Solar panel.**

waves one after another without waiting for the ultrasound to die out. Unfortunately, with a transmission time of 3 ms / bit and a minimum of 8 bits per ultrasound pulse, the length of the ultrasound is almost as long as the lifetime of the pulse we use, but our experience shows that the bit error rate was very high. In addition, ultrasound modulation also increases the power and CPU consumption on the listener.

5.1.3 Solar Power

To minimize the chore of replacing beacon batteries, we considered alternative power sources. Of course, Cricket beacons can be plugged into wall power. We provided an adaptor for this. Beyond this we were interested in renewable power sources, such as solar. Cricket beacons are most often deployed to provide location information indoors, where GPS doesn't work. In this environment, especially office buildings, there is usually a large fluorescent lighting system. We decided to see if we could take advantage of this infrastructure to provide power to the beacons and maintain their easy deployability.

We started with 60mm square solar cells, model OK-60 from OKSolar.com, purchased for about $3.50 each in small quantities. These cells are rated at 3V, 40mA. This level of performance, however, is most likely achieved outdoors on a sunny day. Indoors we typically see 1.4V at 0.2mA and up to 3.2V at 1.7mA near a light fixture, open circuit. A Cricket v1 beacon needs 2.4V at 5mA.

On closer examination of our solar cells we found significant variations among cells: they varied from 0.3V at 0.45mA to 0.6V at 0.6mA under the same lighting conditions. Connecting cells of mixed characteristics in series drives the current down to that of the minimum current of the selected cells. Connecting mixed cells in parallel drives the voltage down to the minimum voltage produced by that collection of cells.

Beyond this, the voltage and current produced were irregular. This led us to consider adding a voltage regulator. This did not succeed due to the high current consumed by the voltage regulator.

This led us to use four solar cells in parallel. This panel provides 2.5V at 7mA and works. We looked at some possibilities for reducing the panel to three cells to make this power supply less bulky. Three cells did not work. Further analysis showed that the beacon draws about 20mA for a fraction of a second during startup. Adding a 4700μF capacitor rated at 16V allowed a panel of three cells to work. Further tests indicated that a 2200μF capacitor rated at 4V would be sufficient. The resulting system is shown in Figure 13.

We have deployed about 30 of these cells and they generally work fairly well. From time to time people turn off the fluorescent lights and, due to the interconnection of the solar panel with the existing beacon layout, the beacons need to be switched on and off to get them going again. We may also go back to a four cell configuration for working in environments with dim lighting. We believe that solar-powered Cricket devices are viable and more convenient that having to periodically replace AA batteries.

With the improved power consumption of Cricket v2 beacons, we can use smaller and flexible solar cells to power them [22].

5.1.4 Monitoring Infrastructure

For the medium-sized deployment made at MIT to support the people locator application, we wrote a monitoring utility to detect beacons with dead batteries. An addition

to the locator software running on each handheld recorded the set of beacons heard in a time window (*e.g.*, 5 minutes) then reported the beacon names to a server. The server recorded the time at which each beacon was last reported. Beacons not heard from for a long period of time where either dead, or no listener had been near that beacon. The latter could be checked by taking a listener to close to the beacon. We used the same infrustructure to measure battery life times for beacons by leaving a listener sitting near the beacon until the beacon's battery died.

5.2 Configuration Management

Originally, we had envisioned that each beacon would send its space and coordinate information on the RF channel. Over time, we found that it was simpler in many cases for a beacon to only send a unique ID, and for the mapping between the ID and the space/coordinate information to be maintained in a central database. In this approach, the listener or host device downloads the database for each building of interest.

5.3 Assisted Coordinate Configuration

Configuring spatial information in the beaon had been easy, but configuring accurate beacon coordinates has been a major bottleneck in deploying Crickets. It is painful to use measuring tape to obtain beacon coordinates manually. Over time, we have developed a BeaconConfig application to automate the beacon configuration process and allow rapid and ad hoc deployment of the Cricket system in any desired location.

BeaconConfig works by using the Cricket listener to collect distance measurements from all the beacons that needs to be configured. The user picks three beacons as references and places the listener underneath each of them to collect distance samples. The references are used to define the origin and the orientation of the coordinate system. After collecting the measurements, the coordinates of each of the reference beacons are immediately known. The listener can then use the three reference coordinates and the collected distance measurements to compute the coordinates for the rest of the beacons.

The BeaconConfig application assumes that all the beacons that need to be configured are within the listener's receiving range. Thus, it is suitable for open-area or single-room deployment.

5.4 Beacon auto-localization

For a large Cricket deployment, it is inconvenient to configure beacon coordinates by manually visiting each of them. Ideally, what we need is an "anchor-free auto localization" algorithm, where beacons measure inter-beacon distances to their neighbors and run a distributed algorithm to compute a coordinate assignment that satisfies the measured distances.

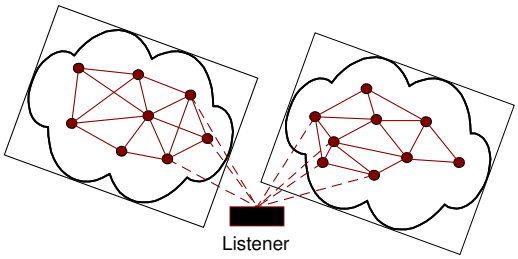

Figure 14: **Mobile-assisted autolocalization: Patching together disconnected beacon regions using a listener.**

A number of recent groups have developed auto-localization protocols for sensor networks. Some of these use a non-trivial number of anchor nodes that are pre-programmed with their positions, and those schemes are not well-suited for Cricket. Anchor-free schemes that are able to handle arbitrary node arrivals match Cricket's requirements better.

We have developed a two-phase fully distributed anchor-free algorithm whose runtime is linear in the number of beacons, for solving this general auto-localization problem. In the first-phase of the algorithm, beacons use the inter-beacon radio connectivity information to come up with an estimated coordinate assignment. This coordinate assignment results in a scaled up approximation of the original graph that maintains the correct ordering of noides in a polar coordinate system. In the second phase, each beacon performs a localized optimization that iteratively reduces the sum-squared-error of the graph [21].

Although this scheme performs well in a general situation where each node knows the distance to its neighbors, most proposed auto-localization schemes (including ours) do not actually solve the configuration problem observed in a real-world Cricket deployment. The reason for this is that mutual distance estimates between beacons are only known when the beacons are in the same open area, for ultrasound does not travel through walls. We have concluded that what is needed is a new *mobile-assisted localization* scheme, where a roving mobile listener "patches together" isolated regions of beacon network (see Figure 14). We are working on such a scheme for Cricket.

6 Related Work

The past few years have seen rapidly growing interest in location-aware applications [4] and in systems such as Active Badge [24], Active Bat [6], Cricket [8], and RADAR [1] that provide location information in indoor environments. Active Badge uses infrared, which has dead spots in some locations, Active Bat and Cricket both use RF and ultrasound like Cricket does, and RADAR uses 802.11 RF. RADAR is not as accurate as the systems that use RF and ultrasound, but does not require any infrastructure other than 802.11 access points.

We do not survey all the location systems here, but only

mention those systems that share some similarities with Cricket. The Bristol indoor positioning system has a design similar to Cricket in that it uses active beacons and passive receivers [9]. The system uses PIC processors, which limits the amount of computation possible on each node in the system. This limit forces the beacons to be placed in a regular pattern on the ceiling, which in turn causes the installation of the beacons to be more difficult.

The Place Lab project uses existing WiFi Access Points (like RADAR) to determine location information for Web applications. Place Lab hopes to provide a community-driven database of WiFi locations for mobile users. Applications can then compare the signal strength of available access points to the Place Lab database to determine a coarse-grained location [10].

Aetherwire & Location and Ubisense are commercial ventures that focus on ultrawideband (UWB) technology to track objects and people. Ubisense reports an accuracy of tens of centimeters in an "active mobile" architecture [2], in which the infrastructure can track users. UWB does not require line-of-sight connectivity, but current systems are not as accurate as Cricket. We also believe that a number of our techniques to handle erroneous distances extend to other systems that do not employ our ranging technology.

Researchers have investigated Bayesian and Kalman filter methods for tracking users [7, 3]. There is an extensive and growing body of work on improving the performance of GPS using various filtering and statistical techniques, some of which might also apply to indoor location systems [12].

7 Conclusion

In the process of our own development and experimentation with Cricket v1, and in cataloguing the reactions of numerous users of the device, we learned a number of useful lessons. First, we learned that the API to the device had to be rich enough to support a wide (and unexpected) variety of usages, and that users did not want all of the underlying mechanism (scheduling, etc.) hidden from them. Second, we learned that applications require either highly accurate location information, or useful fidelity information under degraded conditions, in order to adjust their operation appropriately. We used these lessons to make significant improvements to the distance estimation accuracy, outlier rejection, position accuracy, movement tracking performance, and orientation estimation of Cricket v2. Finally, we learned that deployment and management issues such as configuration and power consumption become significant operational issues even at medium scales, *i.e.*, with the use of just a few tens of devices. We also described how, with our increased understanding of user needs, we modified the capabilities of the beacons, listeners, and the overall system.

We used this knowledge in the design of Cricket v2. Two different prototype units of v2 are now available, and mass-produced units are scheduled to be commercially available in the first quarter of 2004. A noteworthy feature of the new listeners is that they can be attached to a device's CF slot, and can also act as sensor computing nodes with sensors attached to them. Finally, we are also designing a new compass device based on the method described in this paper, and hope to have those units for dissemination sometime in 2004.

References

[1] P. Bahl and V. Padmanabhan. RADAR: An In-Building RF-based User Location and Tracking System. In *Proc. IEEE INFOCOM*, Tel-Aviv, Israel, March 2000.

[2] J. Cadman. Deploying Commercial Location-Aware Systems. In *Proc. Fifth International Conference on Ubiquitous Computing*, October 2003.

[3] D. Fox, J. Hightower, H. Kauz, L. Liao, and D. Patterson. Bayesian Techniques for Location Estimation. In *Proc. Workshop on Location-aware Computing, part of UBICOMP Conf.*, Seattle, WA, October 2003. Available from `http:www.ubicomp.org/ubicomp2003/workshops/locationaware/`.

[4] IT Roadmap to a Geospatial Future. `http://www.nap.edu/html/geospatial_future/`, 2003.

[5] I. Getting. The Global Positioning System. *IEEE Spectrum*, 30(12):36–47, December 1993.

[6] A. Harter, A. Hopper, P. Steggles, A. Ward, and P. Webster. The Anatomy of a Context-Aware Application. In *Proc. 5th ACM MOBICOM Conf.*, Seattle, WA, August 1999.

[7] J. Krumm. Probabilistic Inferencing for Location. In *Proc. Workshop on Location-aware Computing, part of UBICOMP Conf.*, Seattle, WA, October 2003. Available from `http:www.ubicomp.org/ubicomp2003/workshops/locationaware/`.

[8] N. Priyantha, A. Chakraborty, and H. Balakrishnan. The Cricket Location-Support System. In *Proc. 6th ACM MOBICOM Conf.*, Boston, MA, August 2000.

[9] C. Randell and H. Muller. Exploring the Dynamic Measurement of Position. In *Proc. Sixth International Symposium on Wearable Computers*, pages 117–124, Seattle, WA, October 2002.

[10] B. Schilit, A. LaMarca, D. McDonald, J. Tabert, E. Cadag, G. Borriello, and W. Griswold. Bootstrapping the Location-enhanced World Wide Web. In *Proc. Fifth International Conference on Ubiquitous Computing*, October 2003.

[11] M. Spreitzer and M. Theimer. Providing Location Information in a Ubiquitous Computing Environment. In *Proc. 14th ACM SIGOPS Conf.*, pages 270–283, Asheville, NC, December 1993.

[12] G. Strang and K. Borre. *Linear Algebra, Geodesy, and GPS*. Wellesley Cambridge Press, 1997.

[13] Suppressed. Removed for anonymity.

[14] Suppressed. Removed for anonymity.

[15] Suppressed. Removed for anonymity.

[16] Suppressed. Removed for anonymity.
[17] Suppressed. Removed for anonymity.
[18] Suppressed. Removed for anonymity.
[19] Suppressed. Removed for anonymity.
[20] Suppressed. Removed for anonymity.
[21] Suppressed. Removed for anonymity.
[22] Iowa Thin Film home page. `http://www.iowathinfilm.com/`.
[23] `http://webs.cs.berkeley.edu/tos/`.
[24] R. Want, A. Hopper, V. Falcao, and J. Gibbons. The Active Badge Location System. *ACM Transactions on Information Systems*, 10(1):91–102, January 1992.