6th ACM International Conference on Mobile Computing and Networking (ACM MOBICOM), Boston, MA, August 2000
(Slightly revised)

The Cricket Location-Support System

Nissanka B. Priyantha, Anit Chakraborty, and Hari Balakrishnan
MIT Laboratory for Computer Science
Cambridge, MA 02139
{bodhi, achakra, hari}@lcs.mit.edu

Abstract

This paper presents the design, implementation, and evaluation of *Cricket,* a location-support system for in-building, mobile, location-dependent applications. It allows applications running on mobile and static nodes to learn their physical location by using *listeners* that hear and analyze information from *beacons* spread throughout the building. Cricket is the result of several design goals, including user privacy, decentralized administration, network heterogeneity, and low cost. Rather than explicitly tracking user location, Cricket helps devices learn where they are and lets them decide whom to advertise this information to; it does not rely on any centralized management or control and there is no explicit coordination between beacons; it provides information to devices regardless of their type of network connectivity; and each Cricket device is made from off-the-shelf components and costs less than U.S. $10. We describe the randomized algorithm used by beacons to transmit information, the use of concurrent radio and ultrasonic signals to infer distance, the listener inference algorithms to overcome multipath and interference, and practical beacon configuration and positioning techniques that improve accuracy. Our experience with Cricket shows that several location-dependent applications such as in-building active maps and device control can be developed with little effort or manual configuration.

1 Introduction

The emergence of network-enabled devices and the promise of ubiquitous network connectivity has made the development of pervasive computing environments an attractive research goal. A compelling set of applications enabled by these technology trends are context-aware, *location-dependent* ones, which adapt their behavior and user interface to the current location in space, for which they need to know their physical location with some degree of accuracy. We have started seeing the commercial deployment of such applications in outdoor settings (e.g., Hertz's NeverLost navigator on rental cars [13]), where location information is obtained via wide-area technologies like the Global Positioning System (GPS) [11] or using the cellular infrastructure. We believe that the widespread deployment of location-dependent applications *inside* office buildings and homes has the potential to fundamentally change the way we interact with our immediate environment, where computing elements will be "ubiquitous" [20] or "pervasive" [9, 5]. In particular, our work will enable a new class of location-based applications and user interactions in the context of Project Oxygen at MIT [16].

The design and deployment of a system for obtaining location and spatial information in an indoor environment is a challenging task for several reasons, including the preservation of user privacy, administration and management overheads, system scalability, and the harsh nature of indoor wireless channels. The degree of privacy offered by the system is an important deployment consideration, since people often value their privacy highly. The administrative overhead to manage and maintain the hardware and software infrastructure must be minimal because of the potentially large number (possibly several thousands in a building) of devices and networked services that would be part of the system, and the communication protocols must be able to scale to a high spatial density of devices. Finally, indoor environments often contain substantial amounts of metal and other such reflective materials that affect the propagation of radio frequency (RF) signals in non-trivial ways, causing severe multipath effects, dead-spots, noise, and interference.

Our goal is to develop a system that allows applications running on user devices and service nodes to learn their physical location. Once this information is obtained, services advertise themselves to a resource discovery service such as the MIT Intentional Naming System (INS) [3], IETF Service Location Protocol [18], Berkeley Service Discovery Service [8], or Sun's Jini discovery service [14]. User applications do not advertise themselves unless they want to be discovered by others; they learn about services in their vicinity via an active map that is sent from a map server application, and interact with services by constructing queries for services at a required location. By separating the processes of tracking services and obtaining location information, multiple resource discovery systems can be accommodated. By not tracking users and services, user-privacy concerns are adequately met. We emphasize that our goal is a *location-support* system, rather than a conventional *location-tracking* system that tracks and stores location information for services and users in a centrally maintained database.

Over the past many months, we have designed and implemented *Cricket*, a location-support system for building-wide deployment in the context of Project Oxygen, and have conducted several experiments with it. We have integrated it with INS for resource discovery, and an active map application, which together enable location-

This research was supported in part by NTT Corporation, DARPA (Grant No. MDA972-99-1-0014), and IBM.

MobiCom 2000 08/2000 Boston, MA

dependent applications (and users) to discover and interact with services. This paper describes our design goals (later in this section), system architecture and algorithms (Section 2), implementation (Section 3), experimental results (Section 4), applications (Section 5), and a detailed comparison with previous location-tracking systems (Section 6).

The design of Cricket was driven by the following specific goals, which followed from the nature of our applications and from deployment considerations:

- **User privacy.** Whenever a system for providing location information to clients has been deployed in the past, the issue of user privacy has arisen. This is because many previous systems were location *tracking* systems, where a database kept track of the locations of all the entities, including users in the system. To address this concern, we designed a location *support* system, which allows clients to learn their location without centralized tracking in order to construct location-specific queries for resources.
- **Decentralized administration.** Our goal is widespread building-wide deployment. We believe that it is not possible to deploy and administer a system in a scalable way when all control and management functions are centralized. Our design is decentralized—the "owner" of a space in a building (e.g.,the occupant of a room) configures and installs a location *beacon* that announces the identity of that space (some character string) and each beacon seamlessly integrates with the rest of the system. Location receiver hardware, called a listener, is attached to every device of interest to a user. Listeners use an inference algorithm to determine the space in which they are currently located by listening to beacon announcements. There is no need for a central entity to keep track of each individual component in the system.
- **Network heterogeneity.** A wide variety of network technologies exist in most building environments. In our own laboratory, devices and users connected over 10/100 Mbps Ethernet, three different types of indoor wireless LANs, cellular digital packet data (CDPD), infrared, public telephone, and powerline using X10 [21]. Independent of which technology they use to serve or gain access to information, many services and clients can benefit from learning their location in an automatic way, and we would like to accommodate them. In our design, we achieve this by decoupling the Cricket system from other data communication mechanisms.
- **Cost.** Achieving building-wide deployment requires cost-effective components. We use commercial, off-the-shelf, inexpensive components in Cricket, setting and meeting the goal of less than U.S. $10 per location beacon and listener. Our design involves no custom hardware and is small enough to fit in one's palm.
- **Room-sized granularity.** Our goal is a system where spatial regions can be determined to within one or two square feet, so as to distinguish portions of rooms. This requires the ability to demarcate and determine boundaries between regions corresponding to different beacons.

Cricket uses a combination of RF and ultrasound to provide a location-support service to users and applications. Wall- and ceiling-mounted beacons are spread through the building, publishing location information on an RF signal. With each RF advertisement, the beacon transmits a concurrent ultrasonic pulse. The listeners receive these RF and ultrasonic signals, correlate them to each other, estimate distances to the different beacons using the difference in RF and ultrasonic signal propagation times, and therefore infer the space they are currently in. We describe the details of the technologies, the system parameters and configuration, and the algorithms and protocols used in Cricket. The beacons use a decentralized randomized transmission algorithm to minimize collisions and interference amongst each other. The listeners implement a decoding algorithm to overcome the effects of ultrasound multipath and RF interference. We investigate the performance of three decoding algorithms and find that picking the location corresponding to the beacon with minimum statistical mode performs the best, maximizing the likelihood of making the correct choice. We also discuss some practical deployment considerations when using ultrasound hardware, and some location-dependent applications we have developed using Cricket.

2 System architecture

Cricket uses *beacons* to disseminate information about a geographic space to *listeners*. A beacon is a small device attached to some location within the geographic space it advertises. Typically, it is obtained by the "owner" of the location (e.g., the occupant of a room in an office or home, or a building administrator) and placed at an unobtrusive location like a ceiling or wall. Cricket does not attach any semantics to the space advertised by the beacon; any short string can be disseminated, such as the name of a server to contact to learn more about the space or a name resolver for the space to discover resources. Cricket beacons are inexpensive and more than one of them can be used in any space for fault-tolerance and better coverage.

To obtain information about a space, every mobile and static node has a listener attached to it. A listener is a small device that listens to messages from beacons, and uses these messages to infer the space it is currently in. The listener provides an API to programs running on the node that allow them to learn where they are, so that they can use this information to appropriately advertise themselves and their location to a resource discovery service.

The listener can be attached to both static and mobile nodes. For example, when a user attaches a new static service to the network (e.g., a printer), she does not need to configure it with a location or other any attribute; all she does is attach a listener to it. Within a few seconds, the listener infers its current location from the set of beacons it hears, and informs the device software about this via the API. This information can then be used in its own service advertisements. When a mobile computer has a listener attached to it, the listener constantly listens to beacons to infer its location. As the computer (e.g., a hand-held computer carried by a person) moves in a building, the navigation software running on it uses the listener API to update its current location. Then, by sending this information securely to a map server (for example), it can obtain updates to the map displayed to the user. Furthermore, services appear as icons on the map that are a function of the user's current location. The services themselves learn their location information using their own listener devices, avoiding the need for any per-node configuration.

The only configuration required in Cricket is setting the string for a space that is disseminated by a beacon. The specific string is a function of the resource discovery protocol being used, and Cricket allows any one of several possibilities (in Section 5 we describe our implementation platform and integration with INS). Cricket also provides a way by which the owner of a room can securely set and change the space identifier that is sent in the advertisements. This is done by sending a special message over the same RF channel that is used for the advertisements, after authenticating the user via a password. At this stage, we have chosen to allow this change only from within physical proximity of the room or location where the beacon is located. This makes the system somewhat more secure than if we allowed this to be done from afar.

The boundaries between adjacent spaces can either be real, as in a wall separating two rooms, or virtual, as in a non-physical partition used to separate portions of a room. The *precision* of the system is determined by how well the listener can detect the boundary between two spaces, while the *granularity* of the system is the smallest possible size for a geographic space such that boundaries can be detected with a high degree of precision. A third metric, *accuracy* is used to calibrate individual beacons and listeners; it is the degree to which the distance from a beacon, estimated by a listener, matches the true distance. While our experiments show that the distance accuracy of our hardware is smaller than a few inches, what matters is the precision and granularity of the system. These depend on the algorithms and the placement of beacons across boundaries. Our goal is a system with a close-to-100% precision with a granularity of a few feet (a portion of a room).

The rest of this section describes the design of Cricket, focusing on three fundamental issues: (i) mechanism for determining the location (the beacon-listener protocol), (ii) the listener algorithms and techniques for handling beacon interference, and (iii) beacon configuration and positioning.

2.1 Determining the location

At the beginning we were hopeful that a purely RF-based system could be engineered and made to work well, providing location information at the granularity of a room, and ideally, portions of rooms. Our approach attempted to limit the coverage of an RF transmitter to define the granularity of a geographic-space, and using received signal strength to infer best location. Despite many weeks of experimentation and significant tuning, this did not yield satisfactory results [7]. This was mainly because RF propagation within buildings deviates heavily from empirical mathematical models (e.g., see also [6]), and in our environment, the corresponding signal behavior with our inexpensive, off-the-shelf radios was not reproducible across time.

We therefore decided to use a combination of RF and ultrasound hardware to enable a listener to determine the distance to beacons, from which the closest beacon can be more unambiguously inferred. We achieve this by measuring the one-way propagation time of the ultrasonic signals emitted by a beacon, taking advantage of the fact that the speed of sound in air (about 1.13 ft/ms at room temperature) is much smaller than the speed of light (RF) in air. On each transmission, a beacon *concurrently* sends information about the space over RF, together with an ultrasonic pulse. When the listener hears the RF signal, it uses the first few bits as training information and then turns on its ultrasonic receiver. It then listens for the ultrasonic pulse, which will usually arrive a short time later. The listener uses the time difference between the receipt of the first bit of RF information and the ultrasonic signal to determine the distance to the beacon. Of course, the value of the estimated distance is not as important as the decision of which the closest beacon is.

The use of time-of-flight of signals to measure distance is not a new concept. GPS uses the one-way delay of radio waves from satellites to estimate distance, while radio-altimeters in aircrafts use the time for an electromagnetic signal to reflect off the ground to determine altitude. Collision avoidance mechanisms used in robotics [17] determine the distance to obstacles by measuring the time-of-flight of an ultrasonic signal being bounced off them.

It is also possible to measure the distance using the relative velocity of two signals. It is common practice to use the time elapsed between observing a lightning (electromagnetic waves) and accompanied thunder (sound) to estimate the distance to the lightning. The Bat system (detailed in Section 6) uses this idea to determine a mobile transmitter's position in space, where an array of calibrated receivers measure the time of flight of an ultrasonic signal emitted by a mobile transmitter in response to an RF signal from a base station sent to the transmitter and all the receivers.

2.2 Reducing interference

While Cricket has the attractive property that its decentralized beacon network is easy to configure and manage, it comes at the absence of explicit coordination. There is no explicit scheduling or coordination between the transmissions of different beacons that may be in close proximity, and listeners do not transmit any information to avoid compromising privacy. This lack of coordination can cause RF transmissions from different beacons to collide, and may cause a listener to wrongly correlate the RF data of one beacon with the ultrasonic signal of another, yielding false results. Furthermore, ultrasonic reception suffers from severe multipath effects caused by reflections from walls and other objects, and these are orders of magnitude longer in time than RF multipath because of the relatively long propagation time for sound waves in air. In fact, this is one of the reasons it is hard to modulate data on the ultrasonic signal, which makes it a pure pulse. Thus, the listener's task is to gather various RF and ultrasound (US) samples, deduce and correlate the $\{$RF,US$\}$ pairs that were sent concurrently by the different beacons, and choose the space identifier sent from the pair with the closest distance.

We decided not to implement a full-fledged carrier-sense-style channel-access protocol to avoid collisions in order to maintain simplicity and reduce overall energy consumption. Instead, we handle the problem of collisions using *randomization*. Rather than using a fixed or deterministic transmission schedule, beacon transmission times are chosen randomly with a uniform distribution within an interval $[R1, R2]$ms. Thus, the broadcasts of different beacons are statistically independent, which avoids repeated synchronization and prevents persistent collisions. The choice of random interval is governed by the number of beacons we typically expect will be within range of each other and the time it takes for the transmitted information to reach the listeners, which depends on the message size and link bandwidth. In our implementation, we use an average frequency of four times per second distributed in $[150, 350]$ms. A smaller frequency increases the amount of time before a statistically significant location inference can be made, while a higher frequency increases the probability of collisions. We plan to extend this technique to include a listening component that will allow each beacon to infer the number of beacons in its proximity and appropriately scale the beaconing frequency.

We minimize errors due to RF and ultrasonic interference among beacons by two methods: (i) proper selection of system parameters to reduce the chance of false correlations, and (ii) listener inference algorithms based on statistical analysis of correlated {RF,US} samples.

2.2.1 *System parameters*

In addition to transmitting a string corresponding to the space, each beacon transmits a unique identifier. The combination of the location string and identifier is unique across the entire system. This allows the listener to correlate the RF and ultrasonic beacon signals correctly.

The raw line-of-sight range of our ultrasonic transmitter-receiver pair is around 50 feet, when both the transmitter and the receiver are facing each other. However, by mounting the ultrasonic transmitters carefully, as described in Section 3.3, we are able to reduce the effective range to around 30 feet in the absence of any obstacles. The line-of-sight range of the RF transmitter-receiver pair is about 80 feet, which drops to about 40 feet when there is an obstacle (e.g., a wall). Since RF can travel farther than an ultrasonic transmission and can also travel through certain obstacles, it is almost impossible for a listener to receive an ultrasonic signal without receiving the corresponding RF signal.

We discovered that one way to reduce the occurrence of false correlations is to use a relatively *sluggish* RF data transmission rate! Instead, if we used a high-bandwidth RF channel, the data identifying a space would reach a listener before the ultrasound pulse was detected. I.e., if S is the size in bits of the message sent over the RF channel with a transmission rate of b bits/s, and τ is the maximum propagation time for an ultrasonic signal in air between a beacon and a listener, a value of $b < S/\tau$ would mean that the ultrasonic signal corresponding to a given RF message would arrive *while* the S message bits are still being received. Together with the fact that the range of our ultrasound is smaller than our RF, this establishes that *any* potentially correlated ultrasound pulse *must* arrive while an RF message is being received. In the absence of interfering beacon transmissions, this check suffices to do the correct correlation. The specific parameters used in our implementation are described in Section 3.

We now proceed to investigate the different interference scenarios that are possible.

2.2.2 *Interference scenarios*

To better understand the effects of interference and multipath (due to reflected signals) on distance estimation, we characterize the different RF and ultrasonic signals that a listener can hear. Consider the RF and ultrasonic signals sent by a beacon A and an interfering beacon I. The listener potentially hears the following signals:

- RF-A. The RF signal from A.
- US-A. The *direct* ultrasonic signal from A.
- US-RA. The *reflected* ultrasonic signal from A.
- RF-I. The RF signal from I.
- US-I. The *direct* ultrasonic signal from I.
- US-RI. The *reflected* ultrasonic signal from I.

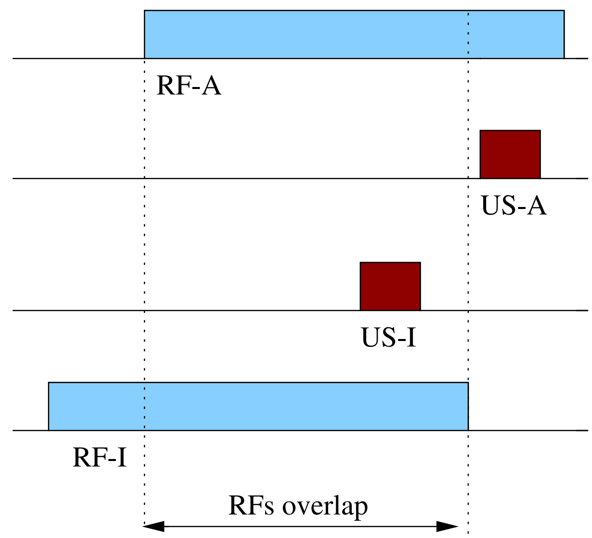

Figure 1: RF-A:US-I interaction, with US-A arriving after US-I. The two RF transmissions overlap in time at the listener.

We only need to consider the cases when a US pulse arrives while some RF signal is being received. The reception of the first ultrasonic signal US-A, US-RA, US-I, or US-RI while RF-A is being received will cause the listener to calculate the distance to A using the time interval between the detection of RF-A and the particular ultrasonic signal. This is because the listener, after receiving the RF signal from a beacon, waits for the first occurrence of an ultrasonic pulse to determine the distance. All subsequent ultrasonic receptions that arrive during this RF message are ignored. Of course, if the direct signal US-A is the first one to be received, the listener correctly estimates the distance to A. However, the wrong correlation of any other ultrasonic signal with RF-A could be problematic.

Case 1: RF-A:US-RA. This combination with the reflected ultrasonic signal from A causes the estimated distance to be larger than the actual distance to A. This situation can occur only if the direct signal US-A was never received by the listener. However, the problems caused by this to the system can be reduced by properly aligned beacons (Section 3.3), as well as using multiple independent beacons per geographic space. In addition, in our experience, we have found that the ability of the ultrasonic waves to bend around obstacle edges (diffraction) makes this a relatively infrequent occurrence since the direct signal is usually detected before the reflected one.

Case 2: RF-A:US-I**.** This is the combination of RF-A with the direct ultrasonic signal from an interfering beacon I, which arrives *before* the ultrasonic signal US-A. Since an ultrasonic pulse can only be received by a listener while the corresponding RF data packet is being received, RF-I should also be in transit to the listener. Hence RF-A and RF-I should overlap at the listener as shown in Figure 1.

If RF-A and RF-I are comparable in signal strength, they will collide, causing the listener to ignore this event because both RF messages will be corrupted. On the other hand, if the signal strength of RF-I is substantially larger than RF-A, the two may not collide and the listener will end up calculating the *correct* distance to beacon I.

The only situation that leads to a wrong distance estimate is when the signal strength of RF-I is much smaller than RF-A, causing the listener to use the RF-A:US-I combination to determine the distance to A. We reduce the chances of this event by using RF signals with longer range than US signals. This generally ensures a strong RF reception whenever the corresponding ultrasonic signal is received (hence the receipt of US-I, in general ensures a strong RF-I).

Case 3. RF-A:US-RI**.** This occurs when a stray reflected signal from an interfering beacon I appears before US-A. As before, this can lead to wrong distance estimates as well.

Although cases 2 and 3 may lead to incorrect distance estimates, our use of randomization reduces the repeated calculation of wrong estimates. If there are a large number of beacons in close proximity to each other, there can be a non-negligible number of wrong distance estimates at the receivers. At this point, we have engineered our system to ensure that there are not more than five or six beacons that are within range of each other at any location.

In addition, listeners do not simply use the first sample pair they get to infer their best location. Rather, they collect multiple samples and use an inference algorithm for this.

2.2.3 Beacon position inference

We develop and compare three simple algorithms to determine which the closest beacon is, overcoming the interference problems of the previous section: *Majority*, *MinMean*, and *MinMode*. In our analysis of these algorithms, the distance estimate is rounded to the nearest ten inches and the data put into different bins according to how frequently they occur. This is done for each beacon separately. Furthermore, isolated stray samples are eliminated from the analysis; a small threshold number of consistent values (two, in our implementation) are needed before the corresponding sample is included for analysis.

- *Majority.* This is the simplest algorithm, which pays no attention to the distance estimates and simply picks the beacon with the highest frequency of occurrence in the data set. This algorithm does not use ultrasonic signals for determining the closest beacon, but as we find in our experiments, this does not perform well. We investigate this primarily for comparison with the other algorithms.
- *MinMean.* Here, the listener calculates the mean distance from each unique beacon for the set of data points within the data set. Then, it selects the beacon with the minimum mean as the closest one. The advantage of this algorithm is that it can be computed with very little state, since a new sample updates the mean in a straightforward way. The problem with this algorithm is that it is not immune to multipath effects that cause the distance estimates to display modal behavior; where computing a statistic like the mean (or median) is not reflective of any actual beacon position.
- *MinMode.* Since the distance estimates often show significant modal behavior due to reflections, our approach to obtaining a highest-likelihood estimate is to compute the per-beacon statistical modes over the past n samples (or time window). For each beacon, the listener then picks the distance corresponding to the mode of the distribution, and uses the beacon that has the minimum distance value from among all the modes. We find that this is robust to stray signals and performs well in both static and mobile cases.

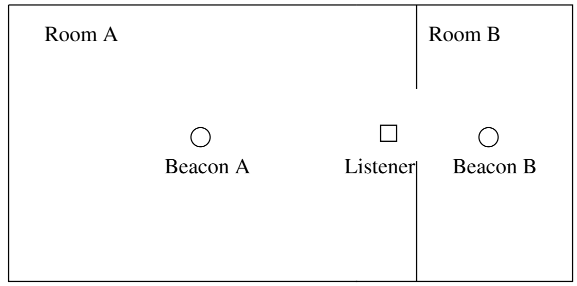

Figure 2: The nearest beacon to a listener may not be in the same geographic space.

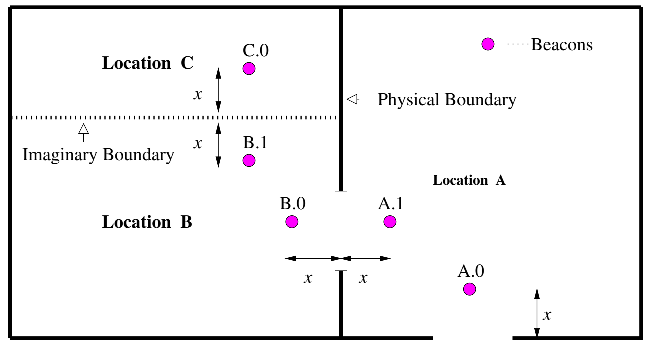

Figure 3: Correct positioning of beacons.

Section 4 discusses the results of our experiments. We note that these are by no means the only possible algorithms, but these are representative of the precision attainable with different degrees of processing at the listeners.

2.3 Beacon positioning and configuration

The positioning of a beacon within a room or space plays a non-trivial role in enabling listeners to make the correct choice of their location. For example, consider the positioning shown in Figure 2. Although the receiver is in Room A, the listener finds the beacon in Room B to be closer and will end up using the space identifier advertised by the latter.

One way of overcoming this is to maintain a centralized repository of the physical locations of each beacon and provide this data to listeners. Systems like the Bat essentially use this type of approach, where the central controller knows where each wall- or ceiling-mounted device is located, but it suffers from two problems that make it unsuitable for us. First, user-privacy is compromised because a listener now needs to make active contact to learn where it is (observe that in Cricket, a listener is completely passive). Second, it requires a centrally managed service, which does not suit our autonomously managed environment particularly well.

Fortunately, there is a simple engineering solution to this problem that preserves privacy and is decentralized. Whenever a beacon is placed to demarcate a physical or virtual boundary corresponding to a different space, it must be placed at a fixed distance away from the boundary demarcating the two spaces. Figure 3 shows an example of this in a setting with both real and virtual boundaries. Such placement ensures that a listener rarely makes a wrong choice, unless caught within a small distance (1 foot in our current implementation) from the boundary between two beacons advertising different spaces. In this case, it is often equally valid to pick either beacon as the closest.

3 Implementation

In this section, we describe the implementation of Cricket. We describe the system parameters and hardware configuration, the API provided by the listener to applications running on the attached node, and some deployment issues with ultrasonic hardware.

3.1 System parameters and hardware

The message size of a beacon RF transmission is 7 bytes long in our implementation, and the RF transmission rate of our radios is 1200 bits/s. It therefore takes about 47 ms for the message to completely reach a listener, during which time an ultrasonic pulse can travel at most about 47 feet. The typical range of our RF radios is about 30 feet in the building. No listener can therefore be farther away than this to detect which space it is in.

Cricket is implemented using inexpensive, off-the-shelf, simple hardware parts that cost less than U.S. $10 per beacon and listener. The beacon consists of a PIC micro-controller running at 10MHz, with 68 bytes of RAM and 1024 words of program memory. It uses a low-power SAW resonator-based RF transmitter and a single-chip RF receiver, both operating in the 418 MHz unlicensed band [10] with amplitude modulation. The final component is an ultrasonic transmitter operating at 40kHz. All of these are assembled on a small board and mounted on a ceiling or high on a wall.

The listener is only slightly more complicated. It has an identical micro-controller, a single-chip RF receiver, and an ultrasonic receiver with a single-chip tone-detector circuit, instead of the corresponding transmitters. It also has a TTL to RS-232 signal converter by which it interfaces to the host device, e.g., a laptop, hand-held computer, or any other service like a printer, camera, television, etc. This interface uses the standard RS-232 protocol at 9600 bits/s.

We measured the power consumption of a beacon, since the periodic transmission of an RF signal and ultrasonic pulse will eventually run the battery down. Although we did not explicitly design the hardware for low power consumption, we find that it is quite efficient, dissipating 15 mW of power during normal operation (when it sends an RF and US signal every 250 ms on average). Currently, each Cricket beacon uses a single 9 Volt re-chargeable battery. We

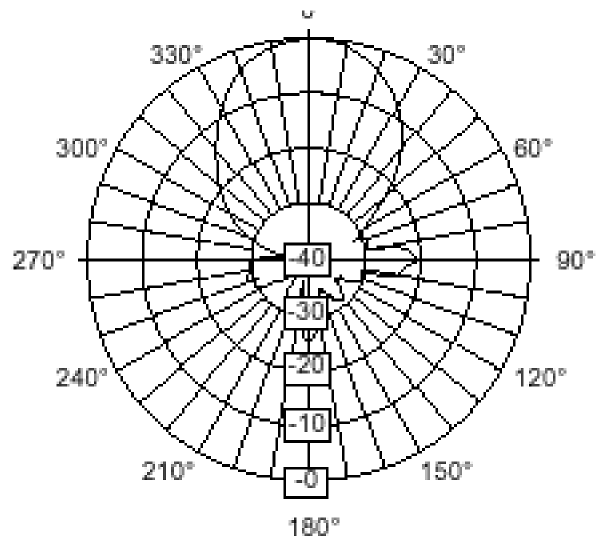

Figure 4: The radiation pattern of an ultrasonic transmitter.

plan to use a solar cell with a backup re-chargeable battery in the future.

3.2 Listener API

A part of the software implemented for receiver nodes, called the LocationManager, runs on the host device that has the listener hardware attached to the serial port. The LocationManager listens on the serial port for any data coming from the listener hardware. In our implementation, the *MinMode* listener inference algorithm to analyze distance estimates is also implemented within the LocationManager, since this provides greater flexibility. The listener sends both the location information and the measured distance to the corresponding beacon, to the LocationManager for each valid RF reception.

Asynchronous to the reception of distance estimates and listener computations, applications running on the host device connect to the LocationManager and retrieve current location information using a datagram socket (UDP) interface. In fact, this allows for the possibility of obtaining this information from a remote node elsewhere on the network, which might be useful for some applications. We have not yet taken advantage of this facility in our applications.

3.3 Ultrasound deployment issues

As described in Section 2, ultrasonic interference at the receiver can lead to incorrect distance measurements. It is therefore important to reduce ultrasonic leakage to other locations while trying to provide full coverage to the location served by a Cricket beacon. We achieve this by proper alignment of the ultrasonic transmitters.

Figure 4 shows the radiation pattern of the ultrasonic transmitter used in the Cricket beacons. This is shown in (r, θ) polar coordinates, where r corresponds to the signal strength in dB and θ corresponds to the offset in degrees from the front of the ultrasonic transmitter. From the radiation pattern, it can be seen that the direction in which the ultrasound transmitter faces (0°) has the maximum signal strength, while the signal strength drops to 1% (-20 dB) of the

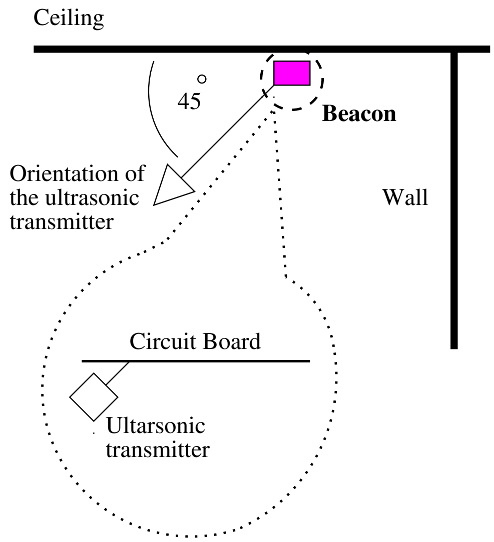

Figure 5: Correct alignment of a Cricket ultrasonic transmitter.

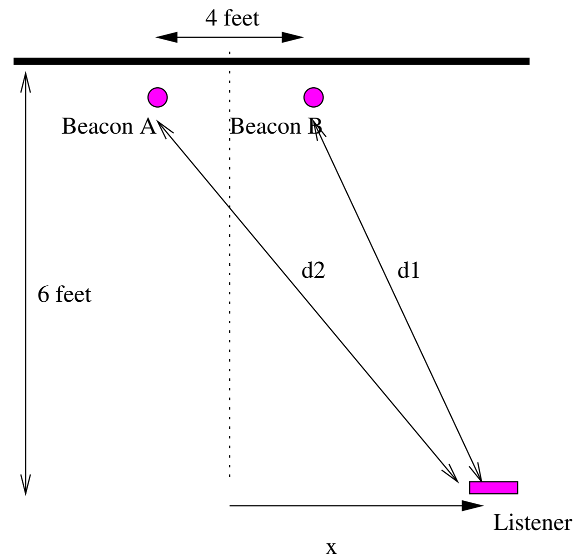

Figure 6: Setup for experiment 1, evaluating boundary performance.

maximum value at $\pm 50^\circ$ away from the 0° direction.

We align the ultrasonic transmitter such that the direction of its peak signal strength is at 45° to the horizontal. The beacon is mounted such that the ultrasonic transmitter faces the location intended to be covered by the beacon. This causes the amount of ultrasonic energy transmitted toward distant locations to be small compared to where it is intended. This alignment is easily accomplished by positioning the transmitter at an angle of 45° to the circuit board of the beacon and mounting the board flat on the ceiling or wall of the room, as shown in Figure 5.

We use the velocity of sound in air to measure distances from beacons to receivers.The velocity of sound depends on environmental factors such as the ambient temperature and humidity. Within a building, these properties can exhibit both temporal and spatial variations. Temporal variations occur at different time-scales such as time of day and season of the year. We avoid errors due to such temporal variations using *relative* rather than absolute distances in determining location (i.e., the primary goal of the listener is to determine which the closest beacon among a set of choices is).

Spatial variations in temperature and humidity due to effects like direct sunlight falling in different sections of a room, the presence of heaters and air conditioners within a room, or the use of humidifiers within a room can affect ultrasound-based distance measurements. We reduce the errors caused by such spatial variations by positioning the beacons and aiming for only relatively coarse-grained location information. For instance, supposing that beacons are always kept 2 feet away from a boundary, the distance recorded from a transmitter in an adjoining room has to decrease by ≈ 4 feet for a receiver to mistakenly assume that the adjoining room is closer. This would require a large variation of temperature and humidity along the path, which is highly unlikely under normal circumstances (the temperature coefficient of the velocity of sound in air is 2 ft/sec per degree-Celsius).

4 Experiments

We conducted several experiments to investigate the performance of Cricket. The first experiment examines the listener performance near location boundaries, and shows that we can achieve a location granularity of 4×4 feet. The second experiment is aimed at investigating the robustness of the system to interference amongst beacons, and the evaluates the performance of the three location inference algorithms presented in Section 2.2.3 for static listeners. The third experiment examines the performance of the three decoding algorithms when a listener is mobile.

4.1 Boundary performance

Figure 6 shows the setup for this experiment. The aim of this experiment is to investigate the the ability of the listener to detect the boundary, which determines the precision of the system.

Two beacons, A and B, advertising different location strings were placed 4 feet apart on the ceiling, giving rise to a virtual boundary in the middle. Distance samples (in the form of ultrasonic pulse propagation time) were taken at 0.5-feet intervals along the x direction as shown in the figure, starting from the center. Figure 7 shows the results of this experiment, plotting the average and the standard deviation of the ultrasonic propagation times from the two beacons as a function of the displacement from the boundary x. This shows that when the listener is more than about 1 foot away from the boundary, the closest beacon can be determined accurately from the estimated distances, thus enabling the listener to determine its location accurately. Furthermore, the difference of the two average distances increases as the listener moves away from the boundary, which causes the probability of making a wrong decision by the listener to decrease as it moves away from the boundary.

This also shows that we can easily achieve a location granularity of 4×4 feet, by placing the beacons in a 4×4 feet grid. Which, effectively divided the region in to 4×4 feet cells. In the future, we

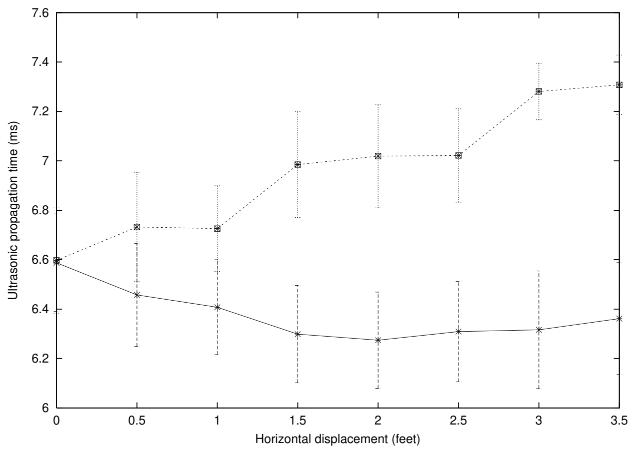

Figure 7: Average and standard deviation (the errorbars) of ultrasonic propagation time as a function of the horizontal displacement of a listener from the boundary of two beacon regions. When the displacement is over about 1 foot, the errorbars do not overlap.

plan to carry out more detailed experiments to measure the accuracy of our hardware, and the precision and granularity of the system as the density of beacons increases.

4.2 Static performance

In the second experiment, we examine the robustness of Cricket against interference amongst nearby beacons. It shows that it is indeed possible to achieve good system performance, despite the absence of any explicit coordination amongst the beacons. We also compare the performance of the three listener inference algorithms presented in Section 2.2.3.

Figure 8 shows the setup for this experiment. Beacons $B1$ and $B2$ provide location information within room X. Beacons $B3$ and $B4$ provide location information for rooms Y and Z. All these beacons are within the range of each others ultrasonic transmissions. To provide RF interference with no corresponding ultrasonic signals (since the range of RF exceeds that of ultrasound in Cricket), we use beacons $I1$ and $I2$ that have their ultrasonic transmitters disabled.

All the beacons were attached to the ceiling with the ultrasonic transmitters facing their respective spaces as described in Section 2.3. We gathered distance samples at locations $R1$ and $R2$ for a static listener. Observe that $R1$ is closer to the interfering sources $I1$ and $I2$ than to the legitimate beacons for the room, corresponding to the presence of severe RF interference. In contrast, $R2$ is only 1 foot away from the boundary separating the rooms X and Y, showing the performance close to a boundary.

Interference Source	I1	I2
Interference at R1	0.0%	0.0%
Interference at R2	0.3%	0.4%

Table 1: Degree of interference at $R1$ and $R2$ caused by $I1$ and $I2$, showing the effectiveness of the randomized beacon transmissions and system parameters.

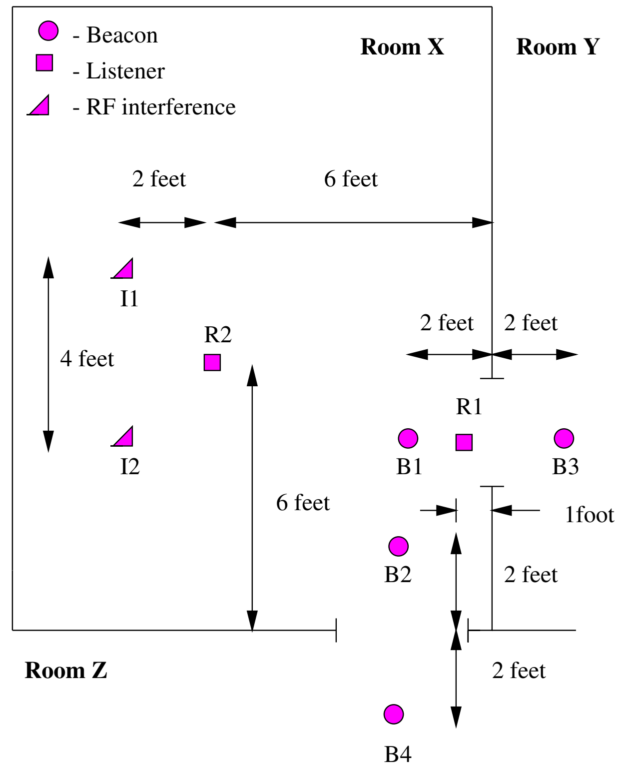

Figure 8: Setup for experiment 2, evaluating the robustness of Cricket in the presence of interfering beacons.

First, we determined the degree of interference caused by $I1$ and $I2$ by collecting 1000 samples of distance estimates at $R1$ and $R2$ and counting the number of values corresponding to each RF source (beacon or interferer). When the listener was at $R1$, somewhat farther from the interfering sources, there were no distance samples corresponding to the interfering RF sources. On the other hand, at $R2$ we received a total of only 7 samples corresponding to both $I1$ and $I2$, despite the fact that $R2$ is closer to $I1$ and $I2$ relative to the legitimate beacons. Table 1 summarizes these results.

The samples corresponding to $I1$ and $I2$ are due to the incorrect correlation of these RF signals with ultrasonic pulses from other beacons in the vicinity of the listener. However, the randomized transmission schedule together with proper system parameters reduces the occurrence of such interference to a very small fraction of the total. This validates our claims in Section 2.2 and our design.

We now investigate the performance of the three inference algorithms, *Majority*, *MinMean*, and *MinMode*, when the listener is at $R1$ and $R2$. Here, we compute the error rate (in percent) in inferring the location by these three inference algorithms, varying the number of distance samples used for inference. The results, shown in Figure 9 (for position $R1$) and Figure 10 (for position $R2$), demonstrate that both *MinMean* and *MinMode* perform very well even when the sample size is small, even for the case when a listener ($R1$) is close to a boundary.

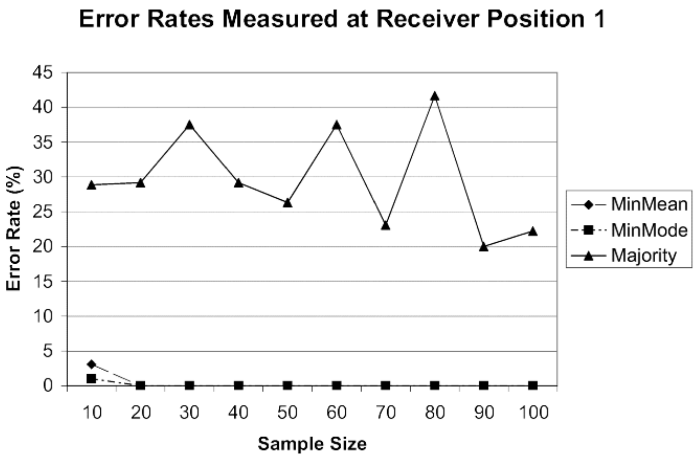

Figure 9: Error rates at Position 1.

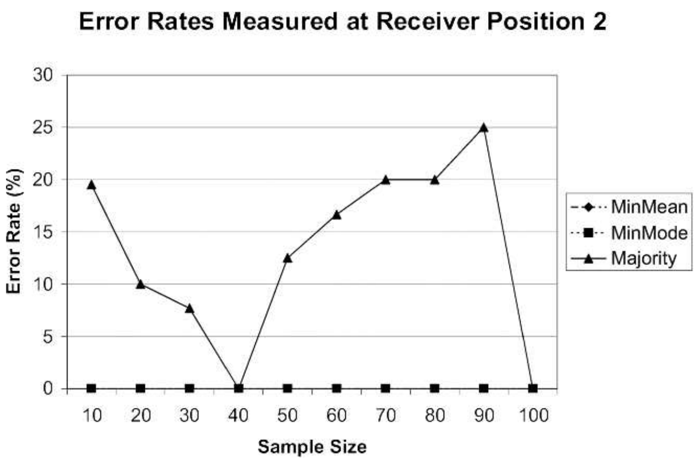

Figure 10: Error rates at Position 2.

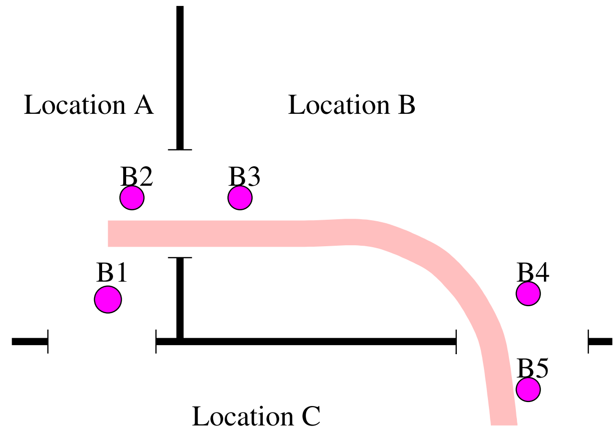

Figure 11: Setup for experiment 3, evaluating the mobile performance of Cricket.

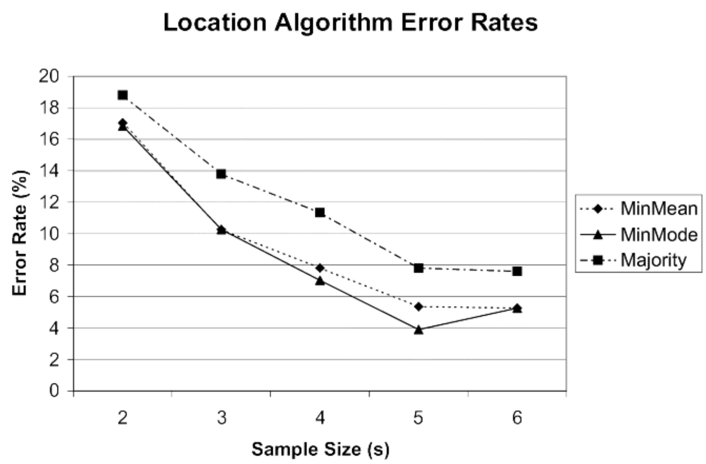

Figure 12: Error rates for a mobile Cricket listener.

4.3 Mobile performance

This experiment is aimed at determining the system performance when the listener is mobile. For a mobile listener, being able to obtain accurate location information within a short time is important. Figure 11 shows the configuration of the beacons and the path followed by the mobile user while taking measurements. The listener was moved through each boundary at approximately the same speed each time, emulating a user's typical walking speed in a building. Each time the listener crossed a boundary, a transition event and a timestamp was logged. Once through the boundary, the listener remained stationary for a short period of time to determine how long it takes to stabilize to the correct value, and then the experiment was repeated again through the next boundary. When analyzing the data, we used the logged transition event to determine the user's actual location with respect to the location being reported by the listener. Note that in this experiment, the listener is always located relatively close to the boundaries.

Figure 12 shows the location error-rate at the listener for the experiment. The error-rate is calculated over the time period during which the listener moves around a location, after crossing a boundary. The *MinMode* performs the best among the three inference algorothms. From the results, it is evident that larger time intervals provide better results over smaller intervals, which is not surprising since a larger interval gives the algorithm more samples samples to work with. Another interesting point is that *MinMean* and *MinMode* both perform about the same over small time windows. As the time interval gets smaller the probability that a distance value sample containing only a single value per beacon increases. A small number of samples causes both the mean and the mode to be the roughly the same.

5 Applications

This section describes how user applications can obtain location information and use it to gain access to nearby services. As mentioned earlier, there are a number of resource discovery systems that can be used along side Cricket. We have implemented several applications using the resource discovery facility provided by the Intentional Naming System (INS), which handles service and device mobility within the naming system [2, 3].

5.1 Using virtual spaces in INS

INS uses the concept of a virtual space (vspace), which is a collection of applications/services that can communicate with each other [15]. Each vspace has a set of name resolvers that resolve name requests for entities in that vspace; each entity is described using an *intentional name*, which is a hierarchical collection of application-defined attributes and values.

The overhead for creating a vspace in INS is small. For our location-dependent applications, we create a vspace for every location of interest (e.g., a room or a floor of a building) and identify it by a string. Each beacon advertises the name of the vspace of the corresponding location, and each listener uses this name to bootstrap into its environment by contacting INS and learning about the other existing services in that vspace.

Users and devices can also register their intentional names with the vspace for that location, which enables other entities in that vspace to detect their presence. This way the user can easily determine all the services that are located in their vspace. A user does not necessarily have to be limited to only one vspace at a time, and can select arbitrary services to use. For example, one vspace can correspond to the set of printers in a building while another corresponds to the services located on a specific floor. A user can determine the least loaded printer by querying the printer vspace, or the physically-closest, least-loaded printer by querying the vspace representing the particular floor of the building.

5.2 Floorplan

The Floorplan is an active map navigation utility that uses Cricket and a map server to present a location-dependent "active" map to the user, highlighting her location on it as she moves. It also displays the set of services that are located in the vicinity of the user, which are dynamically updated as the user moves. Floorplan loads map images from the map server, which also provides the values of (x, y) coordinate on the map corresponding to the user's current vspace position. As the user moves around the building, the listener infers its location and asks the map server to provide the location on the map. Floorplan also learns about various services in the vspace, and contacts those services and downloads a small icon representing each service. These icons are displayed on the map; when the user clicks on an icon, Floorplan uses INS to download a control script or program for the application represented by that icon, and load the controls into a new window so the user can control the application. Figure 13 shows an active map displayed by Floorplan; we see that the user (represented by the dot) is in room 503. It also displays four services it has found in the environment (space) :an MP3 service (represented by the speaker icon) in room 503, a TV service (represented by the TV icon) in room 504, and two printers (represented by the printer icons) in room 517. Using this, a user with no knowledge of her environment or software to control services within it can bootstrap herself with no manual configuration.

6 Related work

There are various solutions available today for device tracking and location discovery. For example, active and passive electromagnetic and optical trackers are sometimes used for tracking and tagging objects. Unfortunately, these tend to be expensive, and the performance of electromagnetic trackers is affected by the presence of metallic objects in the environment. Furthermore, these products do not usually preserve user privacy.

The rest of this section discusses three systems that influenced various aspects of Cricket, and compares their relative benefits and limitations. Table 2 summarizes the following discussion.

6.1 The Bat system

In the BAT system, various objects within the system are tagged by attaching small wireless transmitters. The location of these transmitters are tracked by the system to build a location database of these objects [1, 12].

The system consists of a collection of mobile or fixed wireless transmitters, a matrix of receiver elements, and a central RF base station. The wireless transmitter consists of an RF transceiver, several ultrasonic transmitters, an FPGA, and a microprocessor, and has a unique ID associated with it. The receiver elements consist of an RF receiver, and an interface for a serial data network. The receiver elements are placed on the ceiling of the building, and are connected together by a serial wire network to form a matrix. This network is also connected to a computer, which does all the data analysis for tracking the transmitters.

The RF base station orchestrates the activity of transmitters by periodically broadcasting messages addressed to each of them in turn. A transmitter, upon hearing a message addressed to it, sends out an ultrasound pulse. The receiver elements, which *also* receive the initial RF signal from the base station, determine the time interval between the receipt of the RF signal and the receipt of the corresponding ultrasonic signal, from which they estimate the distance to the transmitter. These distances are then sent to the computer which performs the data analysis. By collecting enough distance readings, it is possible to determine the location of the transmitter with an accuracy of a few centimeters, and these are keyed by transmitter address and stored in the location database.

Bat derives its accuracy from a tightly controlled and centralized architecture that tracks users and objects. In contrast, Cricket is highly decentralized and there is no central control of any aspect of the system, which preserves user privacy, is simpler, and reduces management cost. The differences in design goals between Bat and Cricket lead to radical differences in architecture, although the use of ultrasound and RF is common to both systems.

6.2 The Active Badge system

The Active Badge[1] system was a predecessor to the Bat system, and tracks objects in an environment to store in a centralized location database [19]. Objects are tracked by attaching a badge, which periodically transmits its unique ID using infrared transmitters. Fixed infrared receivers pick up this information and relay it over a wired network. The walls of the room act as a natural boundary to infrared signals, thus enabling a receiver to identify badges within its room. A particular badge is associated with the fixed location of the receiver that hears it.

Like the Bat system, the object tracking nature of Active Badge

[1] Active Badge is a registered trademark of Ing. C. Olivetti & C., S.p.A.

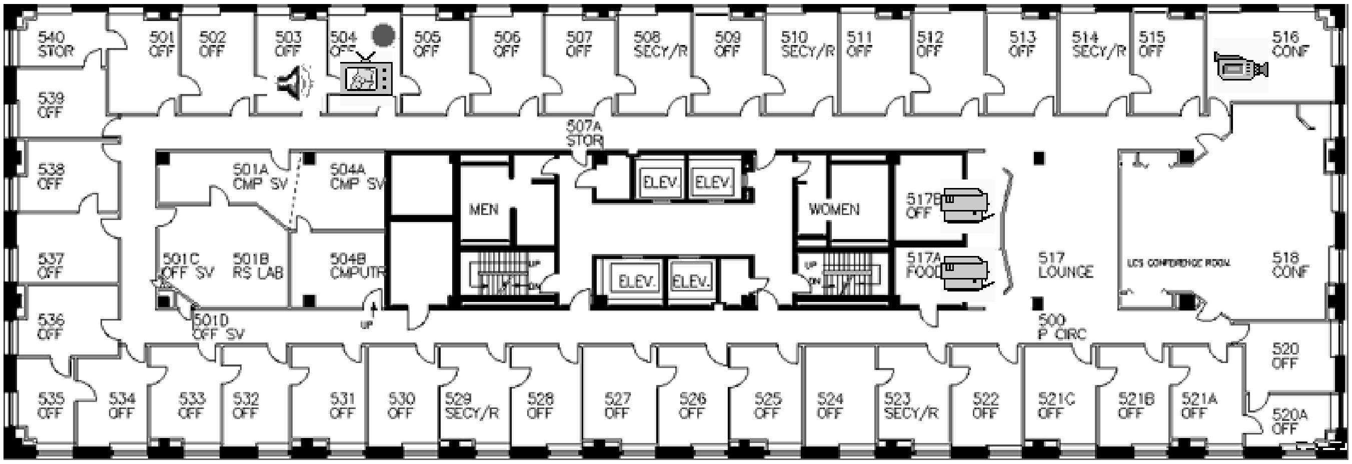

Figure 13: Floorplan map.

System	Bat	Active Badge	RADAR	*Cricket*
User privacy	No	No	Possible, with user computation	*Yes*
Decentralized	No	No	Centralized RF signal database	*Yes*
Heterogeneity of networks	Yes	Yes	No	*Yes*
Cost	High	High	No extra component cost, but only works with one network	*Low (U.S. $10) component cost*
Ease of deployment	Difficult; requires matrix of sensors	Difficult; requires matrix of sensors	RF mapping	*Easy*

Table 2: Qualitative comparison of other location-tracking systems with Cricket.

system may introduce privacy concerns among users. Infrared also suffers from dead-spots, which Cricket and Bat are relatively immune to because they use ultrasound.

6.3 RADAR

The RADAR system implements a location service utilizing the information obtained from an already existing RF data network [4]. It uses the RF signal strength as an indicator of the distance between a transmitter and a receiver. This distance information is then used to locate a user by triangulation.

During an off-line phase; the system builds a data base of RF signal strength at a set of fixed receivers, for known transmitter positions. During the normal operation, the RF signal strength of a transmitter as measured by the set of fixed receivers, is sent to a central computer, which examines the signal-strength database to obtain the best fit for the current transmitter position.

In contrast to these three projects, Cricket has different design goals: it has to handle network heterogeneity and privacy concerns, and have low management cost. It eliminates all central repositories of control or information, leading to an autonomously administered building-wide service via delegation. The beacons advertising location information are self-contained and do not need any infrastructure for communication amongst themselves. Together with the use of inexpensive, off-the-shelf hardware, this makes deployment easy and cost-effective. In summary, Cricket is a *location-support* service, not a location-tracking one.

7 Conclusion

In this paper, we presented the design, implementation, and evaluation of Cricket, a location-support system for mobile, location-dependent applications. Cricket is the result of five design goals: user privacy, decentralized administration, network heterogeneity, low cost, and portion-of-a-room granularity. Its innovative aspects include the use of beacons with combined RF and ultrasound signals in a decentralized, uncoordinated architecture. It uses independent, randomized transmission schedules for its beacons and a receiver decoding algorithm that uses the minimum of modes from different beacons to compute a maximum likelihood estimate of location. We described some deployment considerations based on our preliminary experience with Cricket and presented a comparison with three important past systems, showing that our design goals led to a different design and properties from past systems.

We are encouraged by our experience with Cricket to date and the ease with which location-dependent applications like active map and location-based services can be implemented. We have demonstrated that it is possible to implement a location-support system that maintains user privacy and has no centralized control.

Acknowledgements

We thank William Adjie-Winoto, Dave Andersen, Steve Bauer, Dina Katabi, Jinyang Li, Rodrigo Rodrigues, Xiaowei Yang, and the ACM MOBICOM reviewers for useful comments and suggestions that improved the quality of this paper.

References

[1] A. HARTER AND A. HOPPER. A New Location Technique for the Active Office. *IEEE Personal Communications 4*, 5 (October 1997), 43–47.

[2] ADJIE-WINOTO, W. A Self-Configuring Resolver Architecture for Resource Discovery and Routing in Device Networks. Master's thesis, Massachusetts Institute of Technology, May 2000.

[3] ADJIE-WINOTO, W., SCHWARTZ, E. AND BALAKRISHNAN, H. AND LILLEY, J. The design and implementation of an intentional naming system. In *Proc. ACM Symposium on Operating Systems Principles* (Kiawah Island, SC, Dec. 1999), pp. 186–201.

[4] BAHL, P., AND PADMANABHAN, V. RADAR: An In-Building RF-based User Location and Tracking System. In *Proc. IEEE INFOCOM* (Tel-Aviv, Israel, Mar. 2000).

[5] BANAVAR, G., BECK, J., GLUZBERG, E., MUNSON, J., SUSSMAN, J., AND ZUKOWSKI, D. An Application Model for Pervasive Computing. In *Proc. 6th ACM MOBICOM Conf.* (Boston, MA, Aug. 2000).

[6] BULUSU, N., HEIDEMANN, J., AND ESTRIN, D. GPS-less Low Cost Outdoor Localization For Very Small Devices. Tech. Rep. 00-729, Computer Science Department, University of Southern California, Apr. 2000.

[7] CHAKRABORTY, A. A Distributed Architecture for Mobile, Location-Dependent Applications. Master's thesis, Massachusetts Institute of Technology, May 2000.

[8] CZERWINSKI, S., ZHAO, B., HODES, T., JOSEPH, A., AND KATZ, R. An Architecture for a Secure Service Discovery Service. In *Proc. 5th ACM MOBICOM Conf.* (Seattle, WA, Aug. 1999), pp. 24–35.

[9] DERTOUZOS, M. The Future of Computing. *Scientific American* (Aug. 1999). Available from `http://www.sciam.com/1999/0899issue/0899dertouzos.html`.

[10] FEDERAL COMMUNICATIONS COMMISION. *Understanding the FCC regulations for low-power, non-licensed transmitters*, Feb. 1996.

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

[12] HARTER, A., HOPPER, A., STEGGLES, P., WARD, A., AND WEBSTER, P. The Anatomy of a Context-Aware Application. In *Proc. 5th ACM MOBICOM Conf.* (Seattle, WA, Aug. 1999).

[13] Hertz Services: Hertz NeverLost. `http://www.hertz.com/serv/us/prod_lost.html`, 2000.

[14] Jini (TM). `http://java.sun.com/products/jini/`, 1998.

[15] LILLEY, J. Scalability in an Intentional Naming System. Master's thesis, Massachusetts Institute of Technology, May 2000.

[16] Oxygen home page. `http://oxygen.lcs.mit.edu/`.

[17] Ultrasonics and robotics. `http://www.seattlerobotics.org/encoder/may97/sonar2.html`, May 1997.

[18] VEIZADES, J., GUTTMAN, E., PERKINS, C., AND KAPLAN, S. *Service Location Protocol*, June 1997. RFC 2165 (`http://www.ietf.org/rfc/rfc2165.txt`).

[19] WANT, R., HOPPER, A., FALCAO, V., AND GIBBONS, J. The Active Badge Location System. *ACM Transactions on Information Systems 10*, 1 (January 1992), 91–102.

[20] WEISER, M. The computer for the 21st century. *Scientific American* (September 1991).

[21] X-10 home page. `http://www.x10.com/homepage.htm`.