The Cricket Indoor Location System

by

Nissanka Bodhi Priyantha

S.M. Computer Science, Massachusetts Institute of Technology (2001)
B.S. Electronic Engineering, University of Moratuwa, Sri Lanka (1996)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2005

Author .....
Department of Electrical Engineering and Computer Science
May 19, 2005

Certified by .....
Hari Balakrishnan
Associate Professor of Computer Science and Engineering
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

The Cricket Indoor Location System

by

Nissanka Bodhi Priyantha

Submitted to the Department of Electrical Engineering and Computer Science
on May 19, 2005, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Computer Science and Engineering

Abstract

Indoor environments present opportunities for a rich set of location-aware applications such as navigation tools for humans and robots, interactive virtual games, resource discovery, asset tracking, location-aware sensor networking etc. Typical indoor applications require better accuracy than what current outdoor location systems provide. Outdoor location technologies such as GPS have poor indoor performance because of the harsh nature of indoor environments. Further, typical indoor applications require different types of location information such as physical space, position and orientation.

This dissertation describes the design and implementation of the Cricket indoor location system that provides accurate location in the form of user space, position and orientation to mobile and sensor network applications.

Cricket consists of location *beacons* that are attached to the ceiling of a building, and receivers, called *listeners*, attached to devices that need location. Each beacon periodically transmits its location information in an RF message. At the same time, the beacon also transmits an ultrasonic pulse. The listeners listen to beacon transmissions and measure distances to nearby beacons, and use these distances to compute their own locations. This active-beacon passive-listener architecture is scalable with respect to the number of users, and enables applications that preserve user privacy.

This dissertation describes how Cricket achieves accurate distance measurements between beacons and listeners. Once the beacons are deployed, the MAT and AFL algorithms, described in this dissertation, use measurements taken at a mobile listener to configure the beacons with a coordinate assignment that reflects the beacon layout. This dissertation presents beacon interference avoidance and detection algorithms, as well as outlier rejection algorithms to prevent and filter out outlier distance estimates caused by uncoordinated beacon transmissions.

The Cricket listeners can measure distances with an accuracy of 5 cm. The listeners can detect boundaries with an accuracy of 1 cm. Cricket has a position estimation accuracy of 10 cm and an orientation accuracy of 3 degrees.

Thesis Supervisor: Hari Balakrishnan
Title: Associate Professor of Computer Science and Engineering

To my parents.

Acknowledgments

Hari Balakrishnan has been more than a perfect advisor during my life at MIT—he has been an advisor, mentor, and a good friend. His energy and enthusiasm still continue to amaze me. During the countless discussions I had with him, I always felt that I was his only student, although he had about ten students.

Seth Teller provided much guidance in the research leading to this dissertation and played a key role in shaping the Cricket system. His continued focus on the practical aspects and applications was an important factor in the impact this project has had. I thank him for serving on my committee.

Robert Morris provided valuable feedback during the writing of this dissertation. I thank him for serving on my committee.

Despite my lack of knowledge in the field of computational geometry, Erik Demaine patiently discussed the correctness of some of the theorems described in this dissertation.

I thank John Guttag for providing me financial assistance and advice when I first joined MIT. He has continued to be a great inspiration during my life at MIT.

This dissertation presents joint work with Hari Balakrishnan, Anit Chakraborty, Erik Demaine, Allen Miu, and Seth Teller. A number of people have contributed to the Cricket project. Cricket started from my joint work with Hari Balakrishnan and Anit Chakraborty. Dorothy Curtis, Allen Miu, and Ken Steele subsequently contributed to the development of the Cricket system. John Ankcorn, apart from contributing to the Cricket project, also proofread portions of this dissertation and helped me survive the summers I spent in California. Michel Goraczko handled the TinyOS port of the Cricket firmware, and did an excellent job of responding to user questions, producing (with Hari Balakrishnan and Allen Miu) the user manual, and maintaining the inventory of Cricket hardware. He also always managed to find me enough Crickets whenever I wanted to run a large-scale experiment. David Moore also contributed to the development of cricket firmware. Adam Smith implemented the Kalman filter-based position tracking algorithms in Cricket. Crossbow Technologies manufactured the current version of the Cricket hardware. I thank Roshan Baliga, Nikos Michalakis, Jorge Nogueras, and Kevin Wang for using Cricket for their thesis work and providing valuable feedback. I thank Anant Agarwal, Stephen Garland, and Larry Rudolph for using Cricket in their pervasive computing course at MIT. I also thank many users around the world for building systems and applications using the Cricket platform.

I thank my colleagues Dave Anderson, Magdelena Balaziska, Vladimir Bychkovsky, Nick Feamster, Bret Hull, Kyle Jamieson, Jaeyeon Jung, Stanislav Rost, Eugene Shih, Ali Shoeb, Alex Snoeren, Godfrey Tan, and Michael Walfish in the Networks and Mobile Systems group for providing a fun and exciting working environment. Other friends from the fifth floor of LCS, Steven Bauer, Chandrasekhar Boyapati, Dina Katabi, Jinyang Li, Athicha Muthitacharoen, Xiaowei Yang, and Yan Zhang enriched my life at MIT. I thank my roommate Lik Mui for the countless discussions we had on deep philosophical issues while sharing a room at Ashdown for several years.

The people I met when I first came to MIT made adjusting to a new country easy—in particular, William Adjie-Winoto, Deepak Bansal, Jeremy Lilley, Hariharan Rahul, and Suchitra Raman created a friendly working environment during my early days at MIT.

This work was supported by the National Science Foundation and by the MIT Project Oxygen partnership (Acer Inc., Delta Electronics Inc., HP Corp., NTT Inc., Nokia Research Center, and Philips Research).

I thank my beloved wife Kusala for providing continued support and encouragement during past several years. I thank our baby daughter Dilini for being a string reason to finish this thesis and for sleeping throughout the night since the day she was born.

I dedicate this thesis to my parents who have dedicated their entire life for my education.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Since time immemorial, knowing one's location has been invaluable to humans for outdoor navigation over land, sea, and air. Early navigators used equipment such as the astrolabe, the sextant, and the octant to determine their location with respect to various celestial bodies [10]. In the twentieth century, advances in electronics and telecommunication enabled technologies such as Long Range Navigation (LORAN), Radio Detection And Ranging (RADAR), and Global Positioning System (GPS) for outdoor use [78, 36]. Outdoor location information has enabled applications as diverse as tracking vehicles, logistical planning, locating people, resource discovery, and games [59, 51, 30, 106, 35].

While outdoor location-aware applications are widespread today, our work is motivated by the promise of *indoor* applications that can benefit from knowledge of location. Such applications span a wide range, including:

- **Human and robotic navigation:**

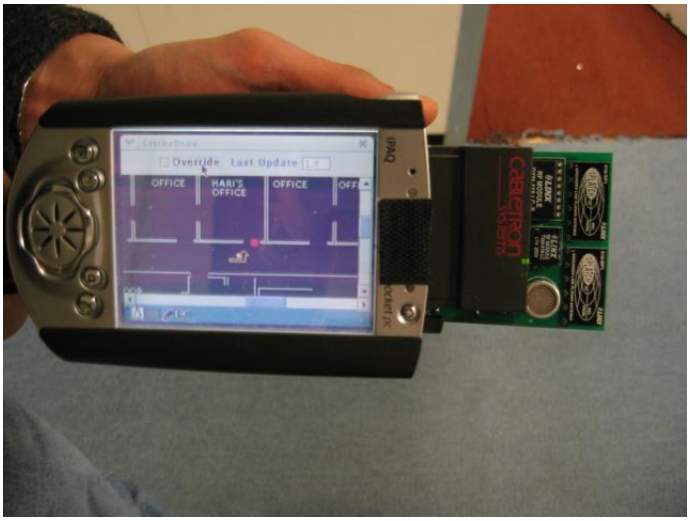


Figure 1-1: A screen shot of a navigation application built using Cricket.

Indoor location information can be used to build tools for navigating in unfamiliar buildings [61], including guiding a traveler to gates in an airport, helping

users navigate (underground) train stations, helping visitors in a museum, directing visitors in an office building, etc. Figure 1-1 shows a screen shot of an indoor navigation application.

In robotic navigation, indoor location information can ease the complexity of robotic path planning. One interesting application enabled by indoor location is robot-assisted elderly care, where a robot uses its own location and the location of people to stay close to the person it is caring for [62].

- **People and object tracking:** Indoor location is also useful for applications that track the location of people inside building. One example is an application that tracks the location of different medical personnel inside a hospital to efficiently assign qualified personnel to various tasks. Other example applications include tracking and reporting the location of people in a user's buddy list, and tracking the location of children in museums or schools [96].

  Object and asset tracking is another category of indoor location-aware applications. Examples include tracking books inside a library, tracking objects inside a warehouse, and tracking assets in an organization. The location of various physical resources such as printers, projects, and copiers also enable resource discovery applications that display the location of various resources in the vicinity of a user [3].

- **Location-enhanced sensor networks:**

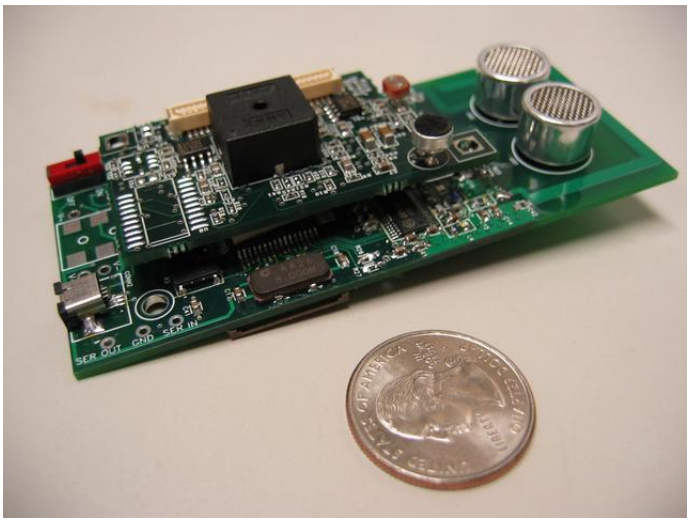


Figure 1-2: A Cricket node with a sensor board attached to it.

Location plays an important role in sensor network systems. In location-enhanced sensing, sensed data such as temperature, humidity, and ambient light level inside a building is annotated with location information 1-2. These location-tagged sensor readings can be used in safety and security applications such as dynamic signage for building evacuation in an emergency, improving

energy efficiency by turning on and off building services such as lighting and temperature based on building occupancy, and efficient building maintenance by quickly locating faulty building services [80, 60]. Location-enhanced sensing can also be used for monitoring conditions in difficult-to-access places of buildings—for example, monitoring air-conditioning ducts for mold growth and monitoring the presence of pests [103].

- **Entertainment:**

Figure 1-3: A screen shot of an interactive version of the Doom game (with permission of Prof. Larry Rudolph.)

Indoor location can be used to build interactive games where physical and virtual worlds overlap. One example is the interactive version of the popular computer game, Doom, built by students taking a pervasive computing course at MIT(Figure 1-3) [98]. Tracking the location of the body parts of a player can be used in interactive video games [13]. Location information is also useful for motion capture to develop animated movies.

- **Human-computer interaction:** When computers become ubiquitous and pervasive, our environments will be filled with computing devices [28]. In such an environment, it would be impractical to interact with each computer using a dedicated display and keyboard. We can use location information to build point and select type user interfaces to interact with these pervasive computing devices [7]. Some examples are the XWand [108] and the Software Flashlight for orientation-aware displays [99]. Another application that requires room-level

location is a video or an audio playback that follows the current location of the user [68].

- **Advertising:** The location of a customer within a shopping mall can be used for sending targeted advertisements, possibly in combination with a map for navigating the mall [1]. Location information can also be used for providing product information inside retail stores [97].

When we examine these applications, we observe that indoor location-aware applications need a higher degree of accuracy, than typical outdoor applications. Most outdoor location systems that exist today are based on RF signals. An indoor environment presents harsh conditions for RF propagation because of the reflections and attenuation caused by various metallic objects. Because of this, traditional outdoor location systems have poor indoor performance.

From the above list of example indoor location-aware applications, we observe that different types of applications need different types of location information. We identify following three types of location information:

- **Space:** Space is defined as a region within some boundary. Examples are rooms and portions of rooms. A room is defined by the walls surrounding it, while a section of a room is defined by a collection of walls and virtual boundaries that are used to demarcate different sections of a room. Space is the most natural form of location information for humans; for example, we often use terms such as "my office" and "living room" refer to different spaces. A space is best denoted by some human-readable text string.

- **Position.** Position is defined as a point in some coordinate system. For instance, the GPS system uses a global spherical coordinate system (latitude, longitude, and altitude) to uniquely identify a position relative to the earth. Building plans, deeds, and city maps all use two-dimensional (2D) coordinate systems to identify the location and the size of different objects. If the position is only used to identify the size of objects and determine how different objects are located with respect to each other, then a local coordinate system with an arbitrary origin can be used. A 2D or 3D local coordinate system can be translated to a global coordinate system if the global and local coordinates of three or four different points are known. We note that the position of some object or a person is not clearly defined because they occupy a region of space rather than a single point. Unless otherwise stated, we define the position of a user or an object as the position of the location sensor carried by the person or the object.

- **Orientation:** Orientation is usually denoted by the angle between a given direction and the *true north*. True north is defined by the meridians of longitude. True north is different from the *magnetic north*, which is the direction a magnetic compass needle points toward. It is also possible to define orientation as the offset between the direction of interest and the coordinate axes, within some

coordinate system. If the coordinate system is a local coordinate system, such as a coordinate system defined within a building, then the orientation information also becomes local. Some indoor applications, such as an application that needs to identify the object a person is pointing at, may be able to use either local or global orientation (the orientation with respect to true north), while some other applications, such as an application that displays a map adjusted to the direction a handheld device is pointing, may need global orientation information. An application for locating objects inside a ship or an aircraft may need only local orientation information. If only local orientation information is known, we can translate it to global orientation if the angle of rotation between the local and the global orientation is known.

This dissertation describes a location system that provides accurate indoor location information in the forms of space, position, and orientation.

1.1 The Cricket System

Figure 1-4: A Cricket hardware unit; this unit can function as either a beacon or a listener.

This dissertation describes the design, implementation, and evaluation of *Cricket*, an indoor location system. The Cricket system consists of Cricket nodes: a small hardware platform consisting of a Radio Frequency (RF) transceiver, a microcontroller, and other associated hardware for generating and receiving ultrasonic signals and interfacing with a host device (Figure 1-4). There are two types of Cricket nodes: *beacons* that act as fixed reference points of the location system and are typically attached to the ceiling and walls of a building, and *listeners* that are attached to fixed and mobile objects that need to determine their location [71]. Beacons periodically transmit an RF message containing beacon specific information—such as a unique beacon-identifier (ID), the beacon's coordinates, the physical space associated with the beacon, etc. Each beacon's transmissions are not centrally coordinated. The listeners listen to beacon transmissions and measure distances to nearby beacons. Each listener uses these distances to determine its space and position; a listener with multiple ultrasonic sensors can use distance difference information to estimate its orientation.

The Cricket system was developed with the following design goals:

- **Small form factor.** We wanted to build a system that can be easily used with mobile devices and sensors. This goal required the hardware implementation to be smaller than the typical handheld device.

- **Accuracy.** Since indoor environments contain closely spaced objects that we may want to uniquely identify by their locations, we wanted a location system that provides a position accurate to a few centimeters.

- **Scalability.** We anticipate that a large number of users and objects will require location information. Hence, we wanted a system that scales with the number and density of users.

- **User privacy.** The user-tracking nature of some previous location systems caused privacy concerns. Our aim was to build a location system that enables a variety of privacy policies, depending on end-user applications.

- **Ease of deployment and configuration.** We wanted to build a location system that is easy to deploy and configure.

1.2 Challenges

Cricket listeners use measured distances to nearby beacons to determine listener location. Since we need accurate location information, the listeners need to be able to measure these distances accurately. This dissertation describes how beacons use a combination of RF and ultrasonic signal transmissions to enable accurate distance measurements at the listener. However, the uncoordinated beacon transmissions can interfere, resulting in incorrect distance samples. This dissertation describes a combination of beacon scheduling algorithms and listener filtering algorithms that prevent incorrect distance samples while maintaining the distributed system architecture.

For a listener to compute its position using measured distances to nearby beacons, the listener needs to know the coordinates of these beacons. To achieve ease of deployment and configuration of the system, we developed algorithms that enable beacons to configure themselves with a coordinate assignment that reflects their true physical layout. These algorithms eliminate the the need for manual coordinate configuration.

The first algorithm produces a set of inter-beacon distances that represents the relative positions of the beacons. However, for a variety of reasons including the line-of-sight requirements of ultrasound-based ranging, it is impossible for the beacons to measure these inter-beacon distances directly.

After inter-beacon distances are obtained, it is necessary to compute a beacon coordinate assignment that satisfies these distances. Since the beacons are deployed indoors, it is impractical to assume the availability of other location technologies, such as GPS, for aiding the beacon coordinate assignment. Hence, Cricket requires a beacon coordinate assignment algorithm that can compute coordinates using only the inter-beacon distances; for obtaining accurate beacon coordinates, this algorithm must tolerate distance measurement errors well.

1.3 Contributions of this Dissertation

This dissertation makes the following contributions. It describes the design, implementation, and evaluation of the hardware, software, and algorithmic components of the Cricket system. The Cricket hardware platform consists of the following major components: an RF transceiver for sending an receiving RF messages, a microcontroller that runs various algorithms and circuitry for transmitting and receiving ultrasonic signals. The software running on Cricket nodes, apart from handling RF and ultrasonic transmissions, also provides an API to set and retrieve different system parameters.

Indoor location systems built prior to Cricket had centralized architectures, where a collection of sensors or transmitters spread across a building were wired to a central controller (see Chapter 2). Cricket was the first indoor location system that used an all-wireless, distributed infrastructure.

Cricket also introduced the importance of location in the form of space for developing location-aware applications. This dissertation describes a beacon deployment strategy that detects boundaries between spaces accurately for determining listener space.

Cricket uses a "passive mobile" approach, where the mobile node passively listens to transmissions from the deployed infrastructure and computes the mobile's position locally. This passive mobile architecture scales well with the number of mobile devices. This architecture also enables applications that preserve user privacy.

Cricket provides all three forms of location information—*space*, *position*, and *orientation*—within a local coordinate system. To our knowledge, Cricket is the only location system that provides all these three types of location information in a small form-factor, within a local frame of reference.

We have developed, implemented, and evaluated the following algorithms in Cricket:

- Beacon scheduling. The distributed Cricket architecture consisting of autonomous beacons with periodic transmissions requires energy-efficient transmission algorithms that avoid interactions between multiple beacon transmissions. We have developed and evaluated a beacon scheduling algorithm that aims to minimize interactions between beacon transmissions.

- Interaction detection. We developed a interaction detection algorithm to detect beacon transmission interactions that may otherwise lead to erroneous distance estimates.

- Outlier rejection. Although the scheduling and interaction detection algorithms can prevent and detect all the inter-beacon interactions under ideal RF propagation models, real-world imperfections such as RF dead-spots cause some beacon interactions resulting in outlier measurement samples at listeners. We have developed two algorithms for filtering out these outliers.

  The *MinMode* algorithm [71], which collects multiple distance samples and selects the measurement with the maximum occurrence, is suitable for static or

slowly moving listeners. For continuously moving listeners, we use a Kalman filter-based approach to detect outliers [92]. In this approach, the listener uses a simplified model to predict its position when it receives a new distance sample, and uses this position estimate to compute an estimated distance to the corresponding beacon. The listener uses the difference between the measured and the estimated distances to reject or accept the new sample.

- Phase difference estimation for orientation. Cricket listeners determine orientation using differential distance estimates. This dissertation presents a unique sensor placement and signal analysis technique that resolves the phase ambiguity problem in phase-difference based differential distance estimation [72].

- Anchor-free localization. Once deployed, Cricket beacons need to be configured with a coordinate assignment that satisfies the physical layout of the beacons. We have developed AFL, an anchor-free distributed localization algorithm, that enables beacons to configure themselves with a valid coordinate assignment [70]. The AFL algorithm runs in two phases. First, it uses RF connectivity to compute an initial coordinate assignment resulting in a beacon a topology that resembles a scaled and unfolded version of the true configuration. In the second phase, it uses inter-beacon distances to run an iterative optimization procedure to minimize the error between the current beacon configuration and the true embedding.

- Mobile-assisted topology generation. The beacon localization algorithm uses inter-beacon distances to compute beacon coordinates. However, obstacles and the characteristics of the ultrasonic sensors used in Cricket prevent beacons from measuring inter-beacon distances directly. The localization algorithm needs a sufficient number of inter-beacon distances such that a coordinate assignment that satisfies these distances correctly reflects the true beacon deployment. We have developed MAT, a mobile-assisted topology generation algorithm to compute a sufficient number of inter-beacon distances using distance measurements at a mobile listener [73].

It must be noted that, although these algorithms were developed and evaluated for the RF and ultrasound-based distance measurement apparatus in Cricket, all the algorithms, except for the beacon scheduling and interaction avoidance, can be used with any ranging technology.

We provide simulation and measurement results on the performance of the individual algorithms and the complete Cricket system. Cricket achieves a distance measurement accuracy of 4-5 cm within a 80^o cone from a given beacon. Cricket achieves a boundary detection accuracy of about 1 cm, a position accuracy of 10-12 cm, and an orientation accuracy of $3^o - 5^o$. The current implementation of Cricket has a form factor which is amenable to be used with handheld devices [24]. Cricket hardware implementation is commercially available from Crossbow technologies [26].

1.4 Organization of the Dissertation

The remainder of this dissertation presents the implementation and evaluation of the Cricket indoor location system. Chapter 2 presents general background and related work on location systems. We present the hardware and software architecture of Cricket in Chapter 3. Chapter 4 describes the RF and ultrasound-based distance measurement in Cricket; this chapter describes algorithms for beacon interaction avoidance, and evaluates the Cricket system scalability performance. Chapter 5 describes the algorithms for obtaining different types of listener location information using distance measurements to beacons. In Chapter 6, we present the MAT algorithm, which computes inter-beacon distances and build a rigid structure for beacon localization, using distance samples at a mobile listener. Chapter 7 describes the AFL algorithm which computes a beacon coordinate assignment using inter-beacon distances obtained from MAT. Finally, Chapter 8 summarizes our work and describe possible future directions for improving Cricket performance.

Chapter 2

Related Work

A location system provides the current location of an object within a given coordinate system. There are two basic approaches to determining the location of an object.

- **Location from landmarks.** In this approach, the location system is implemented by selecting a set of landmarks or reference points with known coordinates. The reference points can be either fixed or moving within the selected coordinate system (e.g. GPS [36]). If the reference points are moving, they should follow a predefined trajectory so that their coordinates can be determined at a given instance. For example, consider three reference points A, B, and C located in a 2D coordinate system. Let d_1, d_2, and d_3 be the measured distances to some object O from these points. If we know the coordinates of the three reference points at the time each measurement was taken, then we can compute the coordinates of O uniquely by solving a system of equations.

  A slightly different form of landmark-based location systems uses boundaries as landmarks. In these systems, boundaries are used to demarcate different *physical spaces*. The boundaries themselves are defined by line segments and curves in some coordinate system. Some examples are: walls that define individual rooms of a building, state-boundaries that identify different states on a map, etc.

- **Location from dead-reckoning.** Dead-reckoning determines the position of an object with respect to some starting point using the dynamics of motion of that object. For instance, if an object O starts moving from some point P along a direction θ at a constant velocity v, its position coordinates at time t are given by $(vt\cos\theta, vt\sin\theta)$. Dead-reckoning relies on the ability of the moving object to accurately measure its dynamics. Since position estimation requires the measured dynamics such as velocity and acceleration to be integrated in time, dead-reckoning suffers from an accumulation of measurement errors. Because of this shortcoming, most location systems are implemented using landmarks, or using a combination of landmarks and dead-reckoning. In the rest of this chapter, we limit our discussion of location systems to landmark-based systems.

This chapter surveys various location systems and the techniques they use to infer location. Section 2.1 gives a general overview of different techniques that can be used to build location systems. Section 2.2 examines different outdoor location systems that exist today. Section 2.3 describes several indoor location systems. Section 2.4 gives an overview of the node localization problem and also describes related work in node localization. Section 2.5 examines different techniques for obtaining orientation information.

2.1 Overview of Techniques to Determine Location

Landmark-based systems need a way by which an object can determine its position based on its proximity to the reference points. The following different approaches may be used to determine the position of an object within a landmark-based system.

- **Distance and angle.** This is the most widely used technique for position estimation. Usually, these distance and angle measurements to the reference points are used to compute the position of the object by triangulation. Outdoor location systems such as GPS [36], RADAR [78], and LORAN [] use this approach to determine location (see Section 2.2).

- **Signal signature.** In this approach, the reference points transmit some signal, usually over RF. The position of an object is determined by measuring the strength of the signals received from one or more reference points. The measured signal strength is used as a *signature* to uniquely identify a given point in space with respect to the landmarks. It is also possible to use an approach where the object transmits and the received signal strength at multiple fixed reference points serve as the signature. Several indoor location systems use this technique to determine a mobile node's position based on the RF signal strength of access points [6, 15, 56, 41, 79, 109].

- **Visibility.** In this approach, an object is associated with a given reference point if that object can receive some signal transmitted by the reference point or visa versa. These systems are usually engineered such that at any given location an object can receive only one reference point's transmission. The Active Badge [105] location system and the CoolTown [55] project use this approach.

2.1.1 Distance Measurement Techniques

Distance measurement is a widely used approach for location estimation. There are three common techniques for measuring the distance to an object from a given reference point:

- **Time-of-flight.** This technique measures the time t taken for some signal to traverse the path between two points (the reference point and the object). If the speed of the signal is v, the distance d is given by $d = v \times t$. For example, GPS uses the time of flight of RF signals to estimate the distance between GPS satellites and the GPS receiver [36].

- **Time-Difference-of-Arrival (TDOA).** TDOA-based schemes measure the distance between given two points using two signals with different speeds that traverse the same path between the two points. Consider two signals A and B with speeds v_A and v_B sent simultaneously by a transmitter. If $v_A > v_B$, then signal B lags behind signal A as they propagate. Let t denote this time lag at a receiver located at a distance d from the transmitter. Then,

$$d = \frac{t}{\frac{1}{v_A} - \frac{1}{v_B}}.$$

  For example, Cricket uses TDOA of RF and ultrasonic signals to measure distance to the reference points.

- **Image processing.** This technique is widely used in passive auto-focus cameras. Here a microprocessor moves a motor-driven lens located in front of a Charge Coupled Device (CCD) sensor array until a focused image of a target is formed on the sensor. Once the image is in focus, for a given lens, the distance between the CCD sensor and the lens l defines the distance d to the target; the distance d can be computed from l and the *F number* of the lens using the laws of geometric optics [45].

2.2 Outdoor Location Systems

Traditionally, location systems were built to support outdoor navigation applications such as navigating military and commercial ships and aircrafts. Most traditional location systems were designed for outdoor applications.

Location information was an invaluable resource for early navigators. These navigators used the Earth as their frame of reference, determining their location by measuring the angle of different celestial bodies relative to the horizon using tools such as the quadrant and the sextant [10]. Celestial bodies act as a collection of natural reference points moving in a predictable path relative to the earth.

The twentieth century saw radical improvements in the quality and accuracy of outdoor location systems. We briefly survey six important systems here: SONAR, VOR, LORAN, RADAR, GPS, and E-911.

2.2.1 Sound Navigation and Ranging (SONAR)

SONAR was developed during the First World War for underwater navigation of submarines. SONAR uses an ultrasonic transmitter to emit ultrasonic pulses, which get

reflected by obstacles such as ships in the sea. The time of flight of these reflected signals is used to compute the distance to the obstacle. Some sonar systems emit an ultrasonic beam that can be rotated either mechanically or using an array of transmitters fed with phase-shifted signals. For a rotated beam, the angle of rotation of the beam determines the angle of the obstacle with respect to the object transmitting the ultrasonic signal. Thus SONAR estimates the position of objects with respect to the coordinate frame defined by the transmitter.

2.2.2 Very High Frequency Omni-directional Range (VOR) System

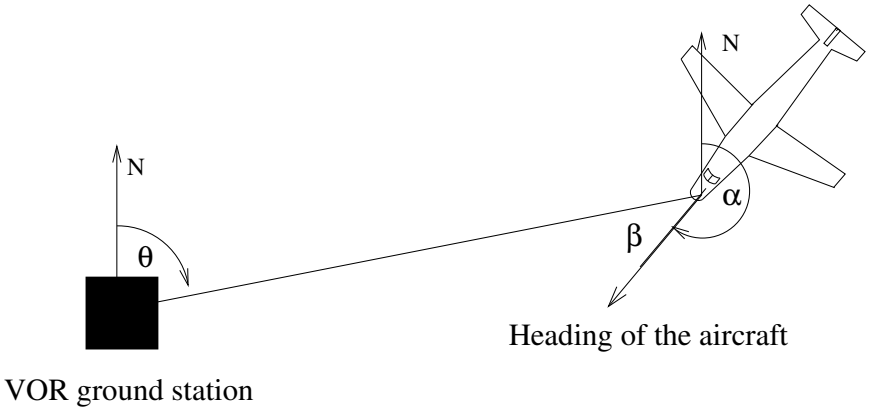


Figure 2-1: Offset β between the aircraft heading and the direction of the VOR station is given by $\beta = \pi + \theta - \alpha$, where θ is the bearing of the aircraft to the VOR ground station and α is the aircraft heading.

During the Second World War, techniques to aid aircraft navigation received much attention. Shortly after the war, the VOR navigation system was developed to provide directional or bearing information of an aircraft with respect to a ground station [46]. The VOR system consists of two components: a VOR ground station, which is usually located close to an airport, and a VOR receiver attached to the aircraft.

Consider the VOR ground station and the aircraft shown in Figure 2-1. Using the VOR system, the aircraft navigation system can compute the angle (bearing), θ, of the aircraft as seen from the VOR ground-station; the navigation system also computes the orientation of the aircraft α using an on-board compass. If the VOR station is co-located with the airport that the aircraft is trying to reach, the angle $\pi + \theta - \alpha$ gives the offset between the aircraft's heading and the aircraft's destination. The pilot uses this offset information to steer the aircraft to its intended destination.

The VOR ground station transmits two simultaneous RF signals (Figure 2-2). The signal S_1 is transmitted from a stationary omni-directional antenna; while S_2 is transmitted from a rotating antenna with two "lobes" as shown in Figure 2-2 (in practice, the rotation is achieved using a circular array of antennas fed by specially modulated signals). Signal S_1 is *frequency modulated* and carries a periodic time-stamp corresponding the time when the +ve lobe of S_2 points toward North. The

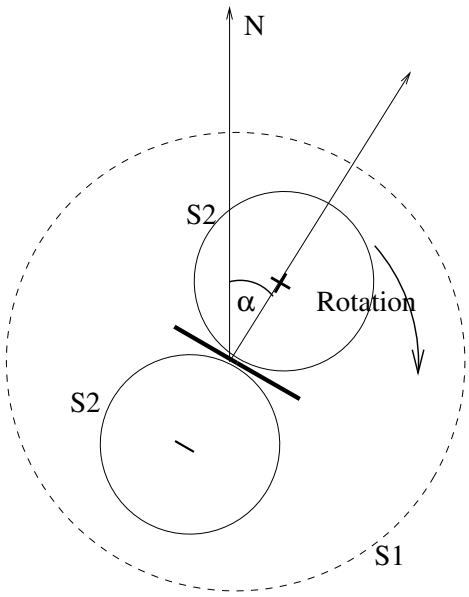


Figure 2-2: A VOR ground station transmits a stationary omni-directional frequency modulated signal S_1 and a rotating directional signal. Aircrafts calculate their bearing to the VOR station by measuring the time shift between a time stamp on S_1 and the peak of S_2.

rotation of S_2 causes it to be *amplitude modulated* when seen by a stationary receiver; moreover, the observed amplitude modulation of S_2 depends on the angle α of the observer with respect to the VOR station. For instance, when $\alpha = 0$, the peak of the modulation on S_2 overlaps with the time stamp on S_1. For a given angle α, the time difference δt between the time-stamp on S_1 and the peak of S_2 is given by:

$$\delta t = \frac{\alpha}{2\pi} \times T,$$

where T is the period of rotation of S_2. Hence, an aircraft can compute its bearing α by demodulating the signals S_1 and S_2 and measuring the time interval δt.

The VOR system implements a location system where an aircraft can measure the angle with respect to VOR stations that act as reference points in a coordinate frame defined by the earth. Some VOR stations also contain a subsystem called *Distance Measurement Equipment* (DME) that enables an aircraft to compute its distance to the VOR station. The DME system composed of a DME transceiver located at the aircraft and a DME transponder located at the ground station. The VOR transceiver periodically transmits RF signals to the DME transponder; the transponder receivers these signals and transmits them back after a known fixed delay. The aircraft transceiver receives the signals from the transponder and uses the time-of-flight of the RF signal to compute the distance to the DME transponder. Hence, a VOR station co-located with a DME provides both the angle and the distance information to the aircraft. Although most of the aircrafts today are equipped with GPS receivers for navigation, VORs are still used for cross-checking the information

from GPS.

2.2.3 Long Range Navigation (LORAN) System

The LORAN system provided guidance information for aircrafts and ships during the latter part of the Second World War. The LORAN system consists of a master RF transmitter and several slave RF transmitters at fixed known locations. The master transmitter periodically transmits an RF signal; the slave transmitters, after a fixed delay from receiving the master transmission, retransmit the same signal. The LORAN receivers, carried by ships and aircrafts, compute the time difference of arrival of signals for each slave and the master.

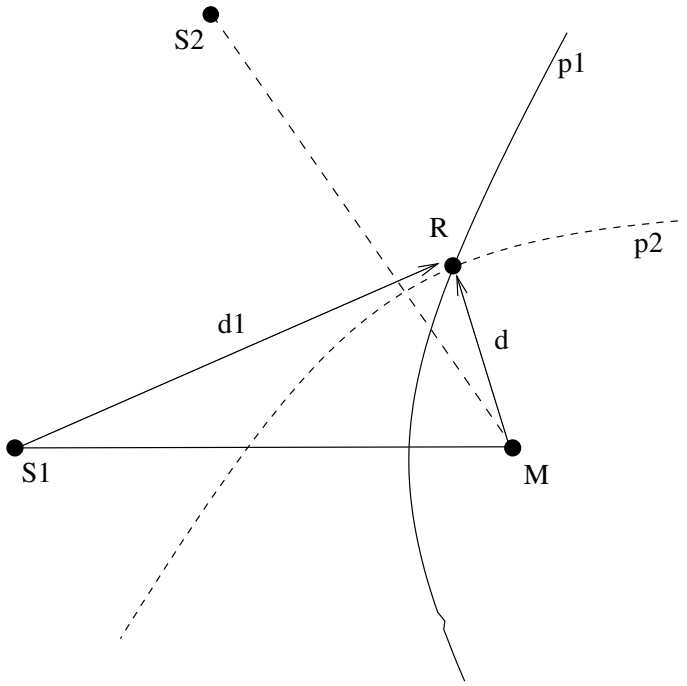


Figure 2-3: Computing the position based on intersecting hyperbolas in the LORAN system.

Consider a receiver R located at a distance d from the master M and distance d_1 from slave S_1 (Figure 2-3). Using the difference of arrival times, the receiver can compute the value $d - d_1$. A given $d - d_1$ value restricts the receiver to be on a hyperbola P_1 with foci M and S_1. The time difference obtained from M and another slave S_2 restricts the receiver to be on the hyperbola P_2. The intersection of P_1 and P_2 gives the location of the receiver. In practice, LORAN uses more than two slave transmitters and uses a maximum likelihood estimate of receiver position to minimize errors due to environmental effects such as storms and RF reflections from the ionosphere.

2.2.4 Aircraft RADAR

RADAR systems are used in diverse applications such as aircraft navigation, detecting enemy aircrafts, detecting speeding vehicles, and predicting weather. The basic RADAR architecture consists of a radio transmitter and a receiver connected to a rotating antenna. The radio transmitter transmits shorts bursts of radio signals, which reflect from objects such as aircrafts, and are received at the radio receiver. The measured time difference between the transmission of a pulse and its reception, ΔT, and the distance D to the reflecting object can be obtained from the equation $D = 0.5c\Delta T$, where c is the speed of the RF signals.

The angle of rotation, θ, of the RADAR antenna, at the time of the reflection, gives the orientation of the object with respect to the earth. Hence the values (D, θ) gives the position of the object (in polar coordinates) with respect to the RADAR antenna on a 2D plane.

2.2.5 Global Positioning System (GPS)

The GPS system consists of twenty seven satellites (as of May 2005) that orbit the earth [36, 48]. Because the satellites follow well-known orbits, their positions are predictable. Each satellite transmits an RF signal encoded with a unique bit pattern. The data streams from different satellites are synchronized using atomic clocks. When a GPS receiver receives signals from multiple satellites, the receiver measures the time shift between the data streams from different satellites. Since the satellite positions are known, the receiver can use the time shifts between signals from any four satellites to solve for the four unknowns that represent the receiver's position in 3D and the current time (the time is treated as an unknown, because the clock of the receiver is not as accurate as the clocks carried by satellites).

Using civilian GPS signals, it is possible to achieve a receiver position accuracies of about fifteen to thirty meters. The main sources of error in receiver position estimation are the additional propagation delays that occur when the signals propagate through the ionosphere and the atmosphere, along the path from the satellites to the receiver. Differential GPS (DGPS) reduces the position estimation error down to about five meters by using GPS signals received at a known position. The GPS receiver at the known position estimates the additional delays caused by ionospheric and atmospheric effects on the signals received from individual satellites, and relay this information to other nearby GPS receivers. These GPS receivers subtract out these additional delays during position estimation [48].

In indoor environments, signals from GPS satellites get attenuated and reflected by various metallic objects [31]. Unlike ionospheric and atmospheric effects that happen at a distance far away from a GPS receiver, effects due to these metallic objects vary rapidly from point to point inside a building. Because of this, it is not possible to use techniques such as DGPS to overcome position estimation errors due to these indoor effects. Thus, indoor GPS performance has fundamental limitations that result in much larger position estimation errors compared to outdoors.

2.2.6 Mobile Phone Location Systems for E-911

The United States Federal Communications Commission's (FCC) E-911 directive [30] requires that a mobile phone operator must be able to provide the location of a user dialing 911.

Some cell phone operators provide this location information using mobile phones equipped with GPS receivers, while other operators use information gathered from the cell phone network to locate callers. The network-based approach uses a combination of RF time of flight and Angle of Arrival (AOA) of mobile phone's signals at the cellular towers to compute the caller's location. The AOA is obtained by comparing the received RF signal strength at multiple antennas on the cellular tower.

2.3 Indoor Location Systems

Traditional location systems such as VOR/DME, LORAN, RADAR, GPS, etc. that provide location information for outdoor navigation are characterized by reference points deployed at known positions. The reference points, in the form of RF ground stations, satellites etc., constitute expensive infrastructure. Most of these systems use RF signals for providing location information, providing a typical accuracy of several meters, which is sufficient for most outdoor applications. Indoor location-aware applications operate in harsher environments that impede RF propagation and these applications often require higher accuracy. On the other hand, indoor applications require a smaller coverage area compared to a typical outdoor system, and it is often desirable to limit the coverage area to a single organization. As a result, several research groups have developed a number of location technologies specifically tailored for indoor applications.

2.3.1 In-building RADAR

The RADAR system developed at Microsoft Research implements a location service utilizing the information obtained from an already existing RF data network [6]. 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.

Many other groups have also developed 802.11-based location systems in recent years [15, 56, 41, 79, 109]. A study has found that such 802.11-based indoor location systems have fundamental limits that result in a median position estimation error of $\simeq 3$ m [32].

2.3.2 The Active Bat Location System

In the Active 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 [42, 43].

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 ultrasonic 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 that 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 decentralized and the infrastructure does not track users or objects, which preserves user privacy. 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.

2.3.3 The Active Badge Location System

The Active Badge system was one of the first indoor location systems. It tracks objects and users and stores their locations in a location database [105]. Objects are tracked by attaching a badge, which periodically transmits its unique ID using an infrared transmitter. Fixed infrared receivers pick up this information and relay it over a wired network to the database. The walls of the room act as a natural boundary to contain 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. The object tracking nature of Active Badge 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.

2.3.4 HiBall Head Tracking System

The HiBall head tracking system, built for precision object tracking in virtual reality applications, uses panels of infrared LEDs that take turns flashing [107]. Several head-mounted cameras measure the position of the flashing LED and the system uses knowledge about the geometry of the head device's cameras to compute the desired location information. The LEDs flash very quickly and thus allow precise information to be obtained. Some of the disadvantages of this system include the difficulty of deploying a large number LED panels to cover an entire building, expensive camera hardware, high computation costs, and the possible interference from the ambient light. Nevertheless, this system provides very precise position information for specialized applications that operate in highly controlled environments.

2.3.5 Ubisense Location System

The UbiSense location system uses Ultra Wide Band(UWB) technology for ranging [101, 104]. This system consists of a small number (*simeq* 4) of UWB base stations and UWB transmitters carried by mobile users, and has a position estimation accuracy on the order of 15 cm. This system uses an active mobile architecture since UWB transmitters are smaller and less expensive compared to UWB receivers currently. Compared to ultrasound, UWB is a better ranging technology, since the RF-based UWB technology enables accurate distance measurements without line-of-sight requirements [18].

2.3.6 Broadband Ultrasonic Location System

The Broadband Ultrasonic Location System was developed as an enhancement to the Cricket location system with increased beacon transmission rate [44]. This system uses broadband ultrasonic transmitters with modulated ultrasonic signals to carry data, this solves the ultrasonic ambiguity (Section 4.2) present in unmodulated ultrasonic ranging systems such as Cricket. However, the wideband modulation scheme requires high ultrasonic transmit energy and costly DSP techniques to demodulate the signals (Girod and Estrin also describe a technique for obtaining robust acoustic ranging by modulating an acoustic signal [37]).

2.4 Node Localization

An indoor location that provides position information needs reference nodes with preconfigured coordinates. The coordinates of these reference nodes can be either local coordinates or they can be global coordinates as in GPS. If these coordinates are local, then the position information within the location system is also local. However, if the global coordinates of 4 (or 3) reference nodes are known, for example using reference nodes with GPS receivers, a 3D (or 2D) local coordinate system can be translated to a global coordinate system. We can use two different approaches for assigning coordinates to a collection of points within a coordinate system.

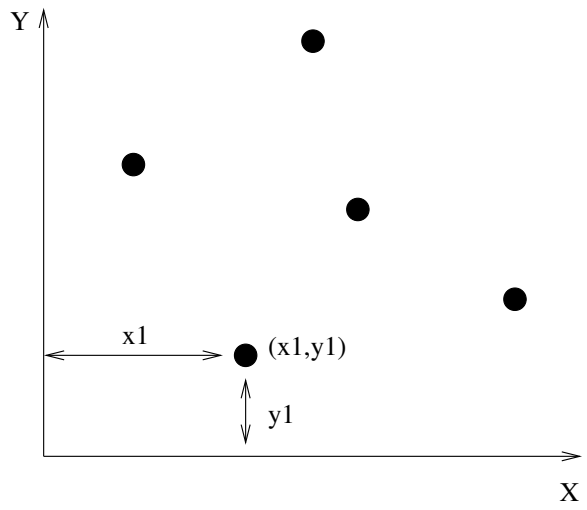


Figure 2-4: Node coordinates can be assigned by measuring the distance from individual coordinate axes.

- **Coordinates from distance measurements to coordinate axis.** After selecting two coordinate axes, we can assign coordinates to reference points by measuring the *distances* to individual nodes from each coordinate axis as shown in Figure 2-4. For example, we can use this technique to assign coordinates to a small number of reference nodes located in a room by measuring the distances to the nodes from two orthogonal walls in the room; here, the two walls become the coordinate axes and the distances determine the coordinates of individual nodes. Although this approach may work well for a small number of nodes in a building with rectangular rooms, such manual configuration becomes tedious when dealing with a large number of nodes. In addition, when all the nodes are not located in the same room, we need an accurate map of the building to obtain a consistent coordinate assignment for nodes in different rooms.

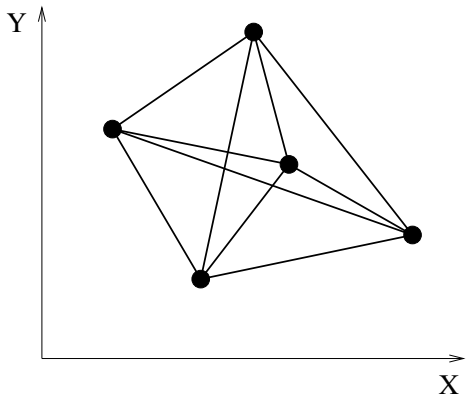


Figure 2-5: Node coordinates can be computed based on inter-node distance measurements.

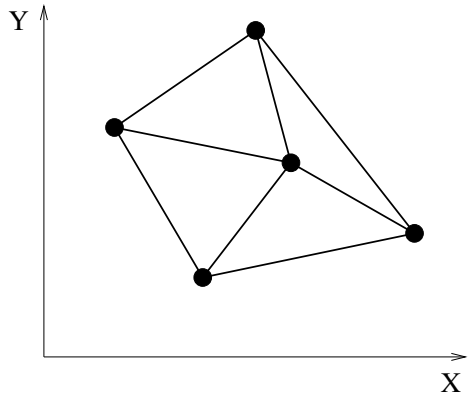


Figure 2-6: Unique relative node coordinates can be computed using only a subset of inter-node distances.

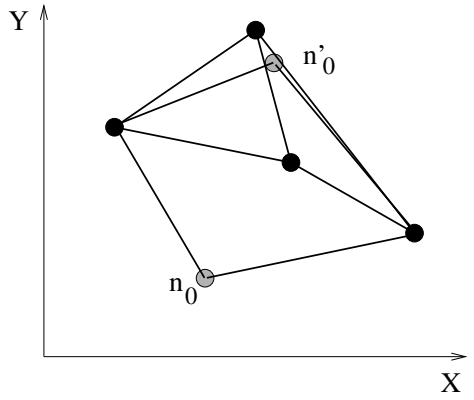


Figure 2-7: Unique relative node coordinates cannot be computed when there are not enough inter-node distances.

- **Coordinates from measurements among nodes.** In this approach we use measurements, such as distances and angles, between pairs of nodes to compute node coordinates. For example, as Figure 2-5 shows, we can use pairwise node distances to compute a coordinate assignment that satisfies these inter-node distances. However, a pairwise distance-based approach does not need *all* the pairwise distances to obtain a node coordinate assignment that represents the node layout. For example, Figure 2-6 has enough pairwise distances such that these distances uniquely determine how nodes are located with respect to each other. We also note that we need a sufficient number of distances to determine the node layout uniquely; for example the distances in Figure 2-7 do not uniquely determine the node layout since there are two possible positions for node n_i with respect to other nodes. Determining enough pairwise distances is an important part in computing a coordinate assignment based on distances which we discuss in Chapter 6.

  It is also possible to use a combination of distances and angles to determine the node coordinates; the set of distances and angles in Figure 2-8 uniquely

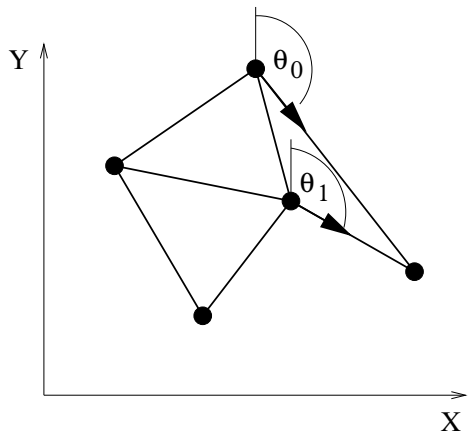


Figure 2-8: Unique relative node coordinates can be computed using a combination of inter-node distances and angles.

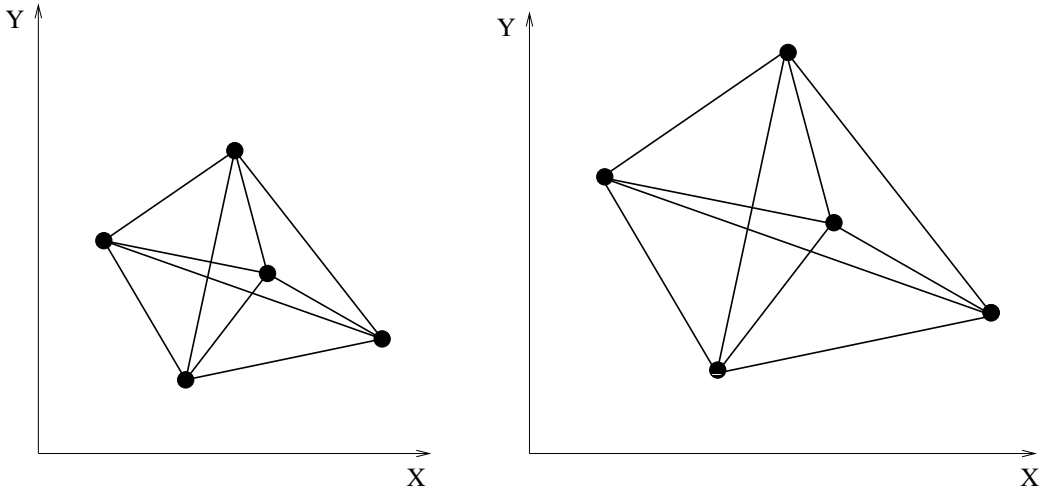


Figure 2-9: Unique relative node coordinates cannot be computed using only angles, since a structure can be scaled up and down while preserving angles.

determine how nodes are located with respect to each other. Unless the location of two or more nodes is known a priori, for example from an external location system such as GPS, an angles-only approach cannot uniquely determine the relative locations of nodes since we can scale a graph up or down while preserving its angles (Figure 2-9).

The inter-node measurement approach does not provide a unique coordinate assignment. There are infinitely many possible coordinate assignments that satisfy the inter-node measurements, since any translation or rotation of a valid coordinate assignment preserves the distances and angles between the nodes. The problem of computing node coordinates using inter-node measurements is called the *node localization problem*, and has received much attention in recent years.

2.4.1 Anchor-based vs Anchor-free Localization Algorithms

Previous research has addressed different versions of the node localization problem. We characterize the algorithms developed to solve this problem in two different ways. The first characterization is based on whether or not the particular algorithm relies on *anchor nodes*, which are nodes with pre-configured coordinates. The second characterization is based on whether the particular algorithm is an *incremental* or a *concurrent* algorithm. Cricket uses an anchor-free, concurrent algorithm (Chapter 7).

- **Anchor-based algorithms.** Algorithms that rely on anchor nodes assume that a certain number or a fraction of the nodes know their coordinates, e.g., by manual configuration or using some other location system such as GPS. The final coordinate assignment of the individual nodes will therefore be with respect some other coordinate system that determined the coordinates of the anchor nodes. A location system built around localization algorithms that need anchor nodes has the limitation that it needs another location system to determine the anchor node positions, and cannot be applied when another location system is unavailable (e.g., inside a building). In practice, Anchor-based algorithms need a large number of anchor nodes for the resulting position errors to be acceptable (see Section 2.4.3).

- **Anchor-free algorithms.** Anchor-free algorithms use local measurements to determine node coordinates, and do not assume the availability of nodes with pre-configured coordinates. For a given graph, a coordinate assignment obtained from an anchor-free localization algorithm will not be unique since the coordinate assignment continues to be valid under translation and rotation.

2.4.2 Incremental vs Concurrent Localization Algorithms

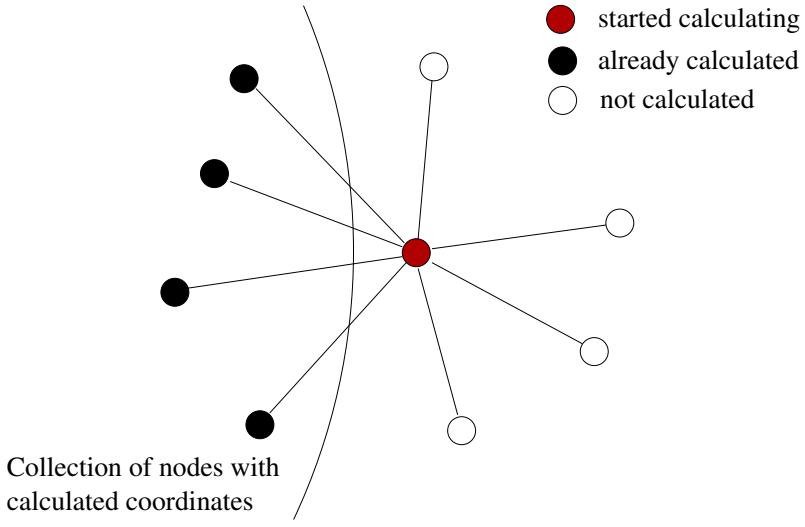


Figure 2-10: Nodes involved in a typical incremental optimization.

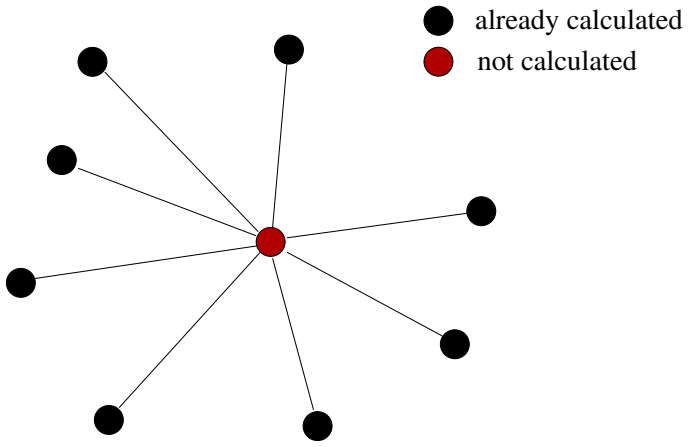


Figure 2-11: Nodes involved in a typical concurrent optimization.

- **Incremental algorithms.** These algorithms usually start with a set of three or four nodes with assigned coordinates. Then they repeatedly add appropriate nodes to this set by calculating the node's coordinates using the measured distances to previous nodes with already computed coordinates. These coordinate calculations are based on either simple trigonometric equations or some local optimization scheme to obtain the best fitting coordinates.

  A drawback of incremental algorithms is that they propagate measurement errors, resulting in poor overall coordinate assignments. Although some incremental approaches usually apply a later global optimization phase to balance such error, it remains difficult to jump out of local minima introduced by the local optimization in the incremental phase.

- **Concurrent algorithms.** In these algorithms, all the nodes calculate and refine their coordinates in parallel. Some of these algorithms use an iterative optimization scheme that reduces the difference between measured distances and the calculated distances based on current coordinate estimates.

  Concurrent optimization schemes have a better chance of avoiding local minima compared to incremental schemes under measurement errors, because they continually balance global error and thereby try to avoid error propagation. For example, consider Figure 2-10, which shows node positions from a typical incremental optimization scheme; in contrast, Figure 2-11 shows the same set of nodes involved in typical concurrent optimization. The layout of the nodes involved in these optimizations more frequently results in an incorrect coordinate assignment (or local minima) for the incremental scheme compared to the concurrent scheme. We present a more thorough experimental comparison in Section 7.6.

2.4.3 Node Localization Algorithms

Moore *et al.* describes a three phase, anchor-free, algorithm to localize a collection of mobile nodes [63]. During the first phase, nodes form clusters and each cluster computes a local coordinate assignment using an incremental algorithm. During the optional second phase, the coordinate assignment within each cluster is improved using a optimization scheme. The third phase of the algorithm stitches the clusters together to obtain a coordinate assignment for all the nodes within some local coordinate system. The use of robust quadrilaterals during the first phase makes this algorithm robust to measurement noise.

The ABC algorithm is an incremental algorithm that does not use anchor nodes [81]. ABC first selects three in-range nodes and assigns them coordinates to satisfy the inter-node distances, and then incrementally calculates the coordinates of nodes using the distances to the three nodes whose coordinates have already been calculated. This incremental scheme results in error propagation. The authors report that with 5% range error, ABC results in about 60% average position error. This sort of cascading error is typical of many incremental algorithms.

Doherty *et al.* describe an anchor-based algorithm for localization using only connectivity constraints among beacons [29]. They represent the connectivities as a set of convex position constraints, and use a centralized linear-programming algorithm to solve for the node positions. Moses *et al.* present a similar approach that uses a maximum-likelihood based centralized algorithm for sensor locations using range estimates to a number of fixed anchor nodes [64].

Bulusu *et al.* describe a GPS-less scheme that uses the radio connectivity of a node to a set of anchor nodes to determine its coordinates [12]. The coordinates of the non-anchor nodes are obtained by calculating the centroid of all the anchors in the nodes radio-range. This is a concurrent algorithm, but it does not use any optimization. In simulations, the authors report about 12% localization error with approximately 12 anchor nodes per non-anchor node ($\frac{R}{d} = 2$ where R is the radio range and d is the separation between anchors). In this scheme, the ratio of the anchor nodes to non-anchor nodes is rather large.

The Terrain algorithm, another anchor-based algorithm, builds on ABC [81]. Each anchor starts the ABC algorithm. Using the coordinates assigned from ABC, each node calculates the distances to at least three anchors. Then, each node performs a concurrent optimization using the distances to the anchors and the anchor coordinates. The authors report about 25% position error (actual offset of the node position from the true position) with 5% range error. They also mention that position errors show a high variance and may diverge during the optimization phase. A related algorithm for localizing nodes in an ad hoc network uses hop-count and radio strength as distance measures, but assumes nearly uniform node density and no occlusion [66].

Savarese *et al.* describe a two-phase, anchor-based, concurrent, localization algorithm [82]. The first phase of the algorithm, Hop-Terrain, is a variant of Terrain, and is robust against ranging errors. The second phase is a simulated-annealing based optimization. With 5% range errors, 10% of the nodes being anchors, and 12 neighbors per node, this algorithm results in about 12% average position error.

Savvides *et al.* describe an anchor-based *collaborative multilateration* [84]. Here, a node solves a set of over-constrained equations relating the distances among a set of anchors and a set of non-anchor nodes (including itself). For a sample graph of 300 nodes, the algorithm needs about 30 (10%) anchor nodes to calculate the location of the other nodes. *Iterative multilateration*, an incremental component of their algorithm, produces node position errors within 20 cm of a node's actual positions, when the ranging error is small (2 cm, Gaussian-distributed). This experiment consists of 50 nodes, with a 3 m ranging system, deployed in a square grid of 15 m $\times$ 15 m, and with 10% of the nodes being anchors.

Niculescu *et al.* present an anchor-based, distributed algorithm that uses *angle-of-arrival* (AOA) for localization [67]. In this algorithm, nodes iteratively obtain position and orientation information starting from anchor (landmark) nodes. One potential problem with this approach is that obtaining precise angle estimates is often difficult.

Howard *et al.*'s localization scheme uses robots equipped with odometric equipment moving through an environment to discover fixed location beacons [49]. Then, a spring-mass based optimization is used to obtain the final beacon coordinates. The authors mention the possibility of the optimization reaching a local minimum, but, also mention that this is a rare event. In Section 7.6.1, we show that spring-mass based optimization schemes with arbitrary initial coordinate assignments have a high probability of reaching a local minimum. We believe that the use of a robot by the authors to explore and discover the beacons resulted in an approximately "order correct" (as explained in Chapter 7) initial coordinate assignment, which in turn reduces the possibility of a local minima during the optimization phase.

Shang *et al.* discuss an anchor free centralized localization algorithm [88]. The authors propose a distributed version of the algorithm that first localizes regions of the network. Later these regions have to be "stitched" together. But the authors do not provide a distributed algorithm for stitching together pieces of the network; we believe that developing such an algorithm is a significant undertaking.

Rao *et al.* use shortest-path hopcounts from a collection of nodes to compute a coordinate assignment that approximates the physical layout of the nodes in a graph [75]. Caruso *et al.* describe a similar algorithm, which uses shortest-path hopcounts from three elected nodes on the periphery of a isotropic graph, bounded by a convex polygon, to compute node coordinates [14].

Bulusu *et al.* study the performance characteristics of different RF-based beacon configuration algorithms and conclude that node density is an important determinant of performance [11]. This paper also contains a detailed survey of various beacon-based localization schemes.

Embedding a graph with given edge lengths also arises in the context of reconstructing the geometry of molecular structures in an area called *distance geometry* [25]. In this context, distance measurements are substantially less accurate, and several techniques have been developed to refine estimates and reduce error bounds by combining several constraints. Berger et al. [8] give efficient algorithms for embedding a graph with error-prone edge lengths, even when nearly half of the edges might have completely inaccurate lengths. However, these algorithms rely on every node having a constant fraction of the nodes as neighbors, for a total of $\Omega(n^2)$ links between

n nodes; this is an unreasonable assumption for a typical indoor node deployment, where the average connectivity is much smaller than $n - 1$.

2.4.4 Mobile-Assisted Node Localization

Once Cricket beacons are deployed, we use a localization algorithm to compute a coordinate assignment that resembles the beacon layout in the space, based on inter-beacon distances. However, the beacons cannot obtain inter-beacon distances due to the properties of the sensors used and the lack of line-of-sight between beacons. Chapter 6 describes how we use a mobile listener to obtain a sufficient number of inter-beacon distances as input to the localization algorithm. Some previous work also use measurements taken with a mobile node to compute the coordinates of a set of fixed nodes.

Scott and Hazas examine different approaches to determine fixed node positions using distance estimates at receiver nodes [87]. Their experiments include both distances obtained at nodes mounted on a mobile frame and raw distances obtained by placing multiple nodes on the floor or from a mobile carried by users. They report better results using the mobile-frame based approach compared to the raw distance approach (however, the paper does not report the size of the fixed frame used). In the raw distance approach, they used simulated annealing to optimize the positions of all the nodes in parallel, this sometimes lead inferior performance due to the presence of local minima in the objective function. In contrast, we break the localization problem to two manageable pieces. We first determine the minimum number of nodes and samples needed per one small group to obtain inter-node distances within this group. Our use of groups with small number of nodes reduces the possibility of local minima and the use of only the computed distances in localization phase makes the localization algorithm scalable with respect to the number of reference nodes.

Pathirana *et al.* uses RF signal strength measurements collected at a mobile robot to localize RF beacons [69]. They use RF signal strength to determine distance between the robot and fixed beacons. The use of the mobile robot improves the accuracy of RF signal strength based distance measurements, since signal strength variations due to spatial fading of RF signals may be reduced. They assume the availability of precise velocity and acceleration of mobile robot to compute the position of the robot within a local coordinate system as the robot collects data. Our approach, although uses data collected at a mobile listener, does not assume any knowledge of the location of the mobile listener during the data collection process.

Corke *et al.* use a flying robot equipped with a GPS receiver to localize stationary nodes [21]. The robot beams down its current GPS coordinates using RF; and the stationary nodes use this information to compute their position. In contrast to our approach, this scheme also assumes that the location of the mobile node is known.

Sichitiu and Ramadurai use a GPS equipped mobile node to localize fixed receiver nodes [90]. They use the RF signal strength to represent the distance between the mobile node and fixed nodes. Again the assumption of availability of mobile node location makes their approach different from our mobile assisted approach.

2.5 Orientation Measurement

Orientation is another form of location information that is useful for developing location-aware applications. The two main techniques for obtaining orientation are: orientation from earth's magnetic field, and orientation from gyroscopes.

2.5.1 Orientation from the Earth's Magnetic Field

The Earth acts as a giant magnet because of the electric currents generated by the movements in its core [38, 93]. The orientation of a given direction can be obtained from the orientation of the Earth's magnetic poles with respect to that direction; this principle forms the basis of operation of the magnetic compass. A freely suspended magnet, in the absence of any stray magnetic fields, aligns itself along the magnetic poles of the Earth. Since ancient times, this property has been used to build magnetic compasses for navigation [2].

Although the location of the earth's magnetic poles are assumed to be fixed (with respect to the earth) for most practical purposes, recent studies have shown that earths magnetic poles are drifting while in some regions the poles have even flipped over [53, 50]. It is also known that the directions of the Earth's magnetic field reverses over a long time scale. The biggest drawback of magnetic compass is the variation of Earth's magnetic field due to ferrous metals such as iron and steel in the environment. The performance of a magnetic compass is also affected by stray magnetic fields caused by various electrical equipment. Because of these shortcomings, they do not perform well in many indoor environments. For instance, when used inside a ship, it is necessary to make corrections to the compass to offset the effects due to ship's steel body. However, such corrections become impractical when the compass is mobile with respect to the surroundings.

A fluxgate compass also determines orientation from the orientation of the earth's magnetic field. However, unlike a mechanical magnetic compass, a fluxgate compass operates by measuring the Earth's horizontal magnetic field using a toroid or a coil.

Similar to fluxgate compasses, electronic compasses use sensors to measure the earth's horizontal magnetic filed. However, instead of a toroid, these use either Hall-effect sensors or Magnetoresistive sensors to measure Earth's magnetic field [47].

2.5.2 Orientation from Gyroscopes

Gyroscopes use a dead-reckoning technique to keep track of the orientation of an object. The gyroscope is first initialized with a known orientation. After the initialization, the gyroscope keeps track of the changes in the object's orientation in 3D space to determine the current orientation of the object with respect to the original direction. Being a dead-reckoning approach, the gyroscopes suffer from accumulation of error.

There are several types of gyroscopes for measuring orientation. A gyrocompass uses a fast spinning wheel mounted on a platform that allows the wheel to freely rotate in 3D. Since a spinning wheel try maintain a fixed orientation, the spinning

wheel retains its original orientation as the object containing the platform rotates in 3D. Since the spinning wheel maintains its original orientation, the objects orientation can be obtained from the offset between the current orientations of the object and the spinning wheel.

In fiber-optic gyroscope, light emitting from a laser diode travels along two fiber-optic cables wound in opposite directions. The phase difference between the light exiting the fiber-optic cables depends on the distances the light traveled along the two cables. When the two windings are rotated around their axis, the distances the light travel along the two windings change, and the angle of rotation reflects itself as a change in the measured phase difference. To determine rotation in 3D, three sets of windings, placed perpendicular to each, are used.

2.6 Chapter Summary

This chapter examined the general principles underlying the design of location systems. This chapter also described various indoor and outdoor location systems, related work on node localization, and systems for obtaining orientation information. The next chapter describes the architecture of the Cricket system.

Chapter 3

Cricket System Architecture

This chapter describes the architecture of the Cricket system. Section 3.1 describes the different components of the Cricket architecture and how they inter-operate. Section 3.2 describes the main software components of the Cricket system. Section 3.3 describes the Cricket hardware design and implementation.

3.1 Cricket System Architecture

Each node in the Cricket system is a small hardware platform consisting of an RF transceiver, a microcontroller, and other associated hardware for generating and receiving ultrasonic signals and interfacing with a host device (Figure 1-4). There are two types of Cricket nodes: beacons and listeners. Cricket beacons act as fixed reference points of the location system and are typically attached to the ceiling and walls of a building, while Cricket listeners are attached to objects that need to obtain their location. Each beacon periodically transmits a radio frequency (RF) message containing beacon-specific information, such as the unique beacon-ID, the beacon coordinates, the physical space associated with the beacon, etc. At the start of the RF message, a beacon transmits a narrow ultrasonic (US) pulse that enables listeners to measure the distances to the beacons using the difference of arrival times of RF and ultrasonic signals. To reduce beacon power consumption and ultrasonic hardware complexity, this ultrasonic pulse does not carry any data. Cricket listeners passively listen to beacon transmissions and compute distances to nearby beacons. Each listener uses these distances and the information contained in the beacon RF messages to compute their space position and orientation (or some subset of this location information).

Each beacon is configured with its "space", a human-readable text string. When beacons are deployed, they do not know their position coordinates. To compute beacon coordinates, a Cricket listener attached to a mobile platform roams around and collects distances from the beacons to itself. Using these distances, a host attached to the listener computes inter-beacon distances; the roaming mobile platform collects enough distances such that the set of computed inter-beacon distances uniquely define how the beacons are located with respect to each other. Next, the host uses these

inter-beacon distances to compute a beacon coordinate assignment that resembles the true beacon embedding.

Distance estimation using coupled RF and US signals enables accurate measurement of beacon-to-listener distances. Since the listeners do not actively transmit messages, the performance of the system is independent of the number of listeners in the environment. As a result Cricket scales well with respect to the number of users and objects that need location information. Since listeners only passively listen to beacon transmissions to determine their location, the position of a user carrying a listener is not tracked by the Cricket system, which enables location-aware applications without compromising user privacy; at the same time, it is possible to build tracking applications to track the position of a rapidly moving object by "inverting" the roles of the beacons and the listeners. Since a beacon coordinate assignment can be computed with only limited manual intervention, the Cricket system is easy to deploy and configure.

3.2 Cricket Software Architecture

The Cricket software architecture consists of two parts: the software running on the attached host, and the firmware running on the Cricket beacon and listener.

3.2.1 Cricket Firmware

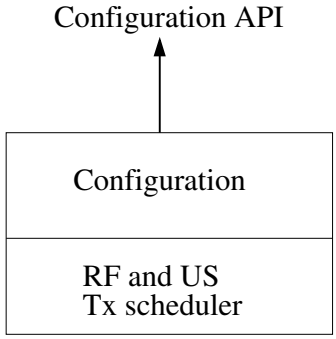


Figure 3-1: Cricket beacon firmware block diagram.

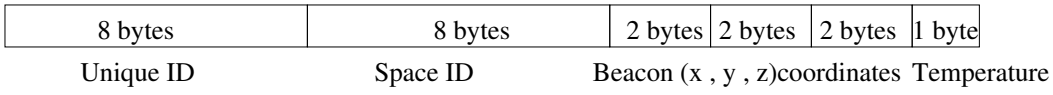


Figure 3-2: The Cricket beacon message format.

Figure 3-1 shows the block diagram of the Cricket beacon firmware. One software module on the beacons schedules beacon RF and ultrasonic message transmissions, while another module exposes an configuration API over the serial RS 232 interface. This API is used to set and read various parameters such as the average beaconing

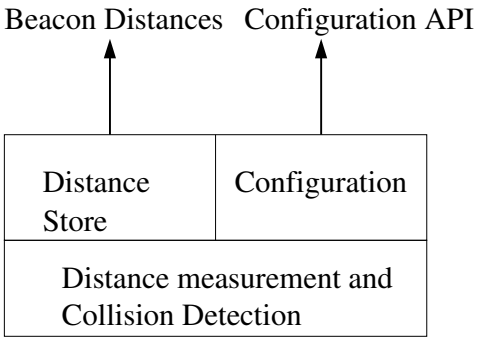


Figure 3-3: Cricket listener firmware block diagram.

rate. The beacon RF message contains the following information: a unique identifier, a space identifier (the name of the space the beacon is located in), beacon coordinates, and the measured ambient temperature at the beacon (Figure 3-2).

The listener firmware consists of three modules as shown in Figure 3-3. The listener distance measurement module uses the timing of RF and US arrival events to measure distances to nearby beacons and also implements the interference detection algorithm described in section 4.1. The listener also contains a configuration module, which exposes an API over the serial RS 232 interface, similar to the beacon. The *distance store* submodule keep track of recently heard beacons and the distances to these beacons; this submodule also implements the "MinMode" algorithm described in section 4.1. The distance store module exposes an API that reports distances to nearby beacons according to the current listener configuration.

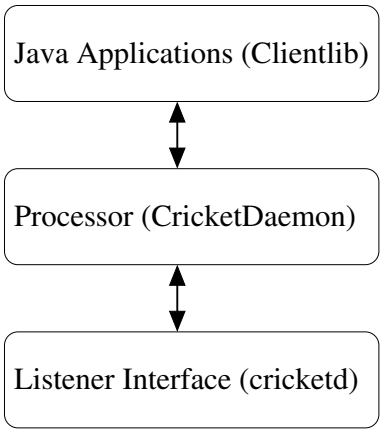


Figure 3-4: Cricket host software architecture.

3.2.2 Cricket Serial Configuration API

The serial configuration API enables the reading and setting of the values of various parameters in the beacons and listeners. This API is inspired by the Hayes AT command set for modems [4]. To issue these commands, the beacon or the listener

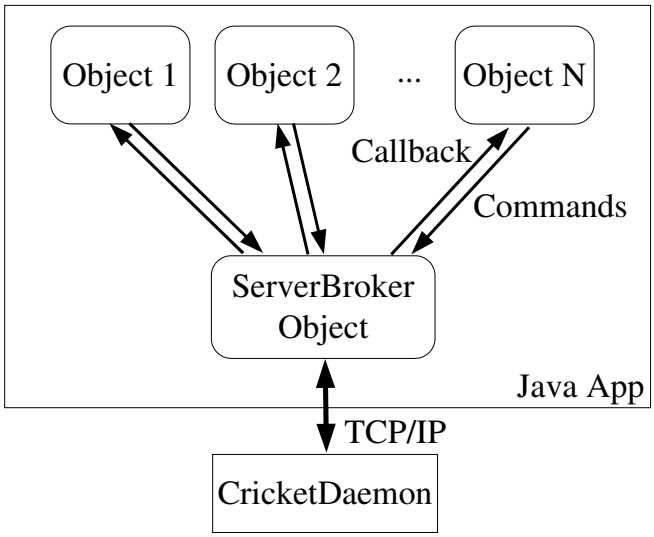


Figure 3-5: Clientlib Architecture.

must be attached to an RS232 serial interface configured as follows:

Transmission speed	115200 bits/second
Data format	8 bits, no parity
Flow control	Xon/Xoff ("software")
Stop bits	1

Once the beacon or the listener is attached to a host, these commands can be issued using a standard serial port utility such as `HyperTerminal` or `minicom`. The command format is:

```
<directive> <command> <parameters>
```

⟨`directive`⟩ "G" for "get" and "P" for "put".
<`command`> The command.
<`parameters`> The argument(s) to the command.

In response to a command, the Cricket listener or beacon echoes the command followed by the result:

```
<command> <result>
```

A complete description of the commands and their parameters are described in the Cricket user manual available from the Cricket Web site (http://cricket.csail.mit.edu).

3.2.3 Cricket Host Software

Figure 3-4 shows the block diagram of the Cricket host software that runs on the attached host. The `CricketD` daemon connects to the serial port and exposes the Cricket serial port API described above. This daemon listens on a TCP port (2947), and applications can access the Cricket serial API by opening a TCP connection to this port. The `CricketDaemon` connects to `CricketD` using a TCP connection. `CricketDaemon` processes raw distance data from `CricketD` and computes the listener space and position. Applications can access `CricketDaemon` through a TCP port (5001). The `Clientlib` Java library provides support to build location-aware Java applications (Figure 3-5). This library is composed of a `ServerBroker` object that connects the `CricketDaemon` to obtain current location information. The `ServerBroker` exports a callback interface. Various Java objects register with the `ServerBroker` to be notified whenever the device's location changes. During registration, these objects use a bit-mask to specify which specific location change events they are interested in (e.g. the current space, position, etc.).

3.3 Hardware Implementation

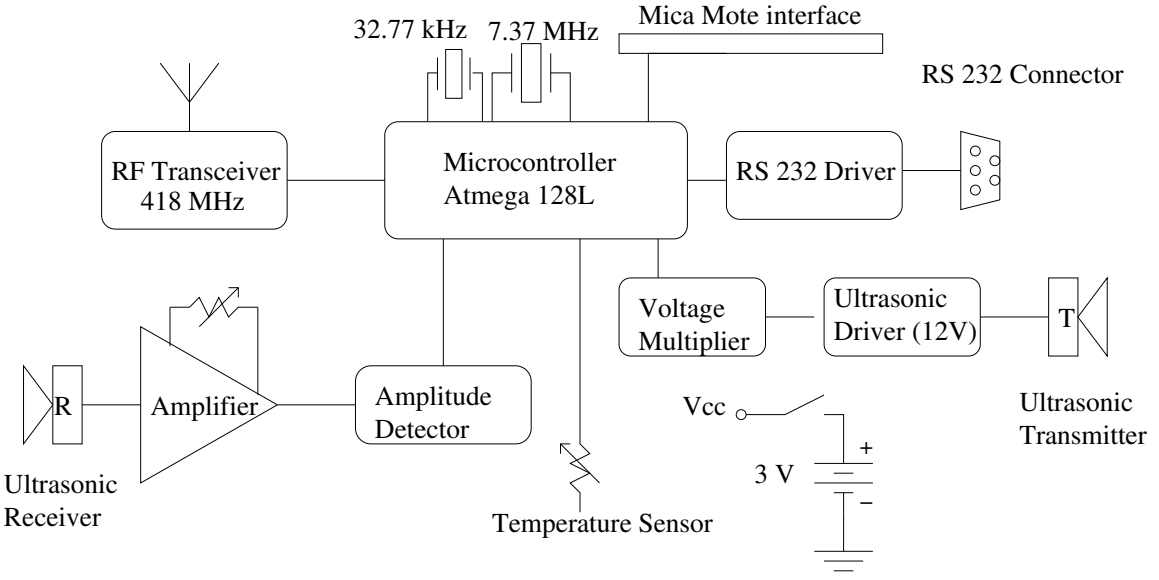


Figure 3-6: Cricket hardware implementation. The beacon and the listener devices are identical.

The hardware implementations of the Cricket beacon and listener are identical. Figure 3-6 shows a block diagram of the Cricket beacon and listener hardware. Figure 3-7 shows the placement of the important components in the hardware implementation.

The Cricket hardware implementation has the following submodules (a complete hardware schematic is in Appendix A).

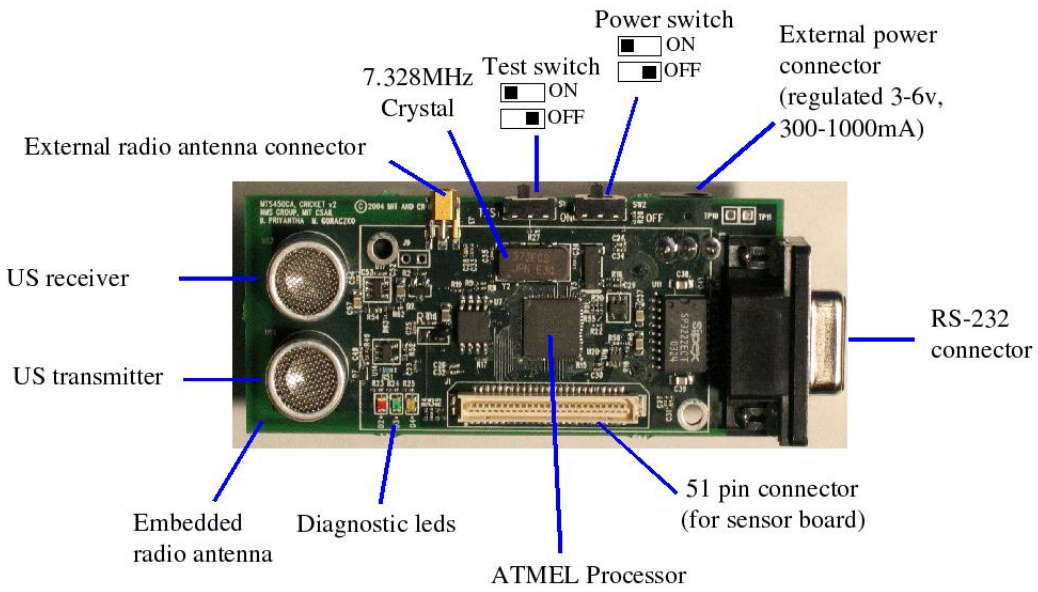


Figure 3-7: Cricket hardware components and layout.

- **Microcontroller.** Cricket uses an Atmega 128L microcontroller operating at 7.3728 Mhz when active [5]. The microcontroller uses a 32.768 kHz clock for timing during the sleep mode. The Atmega128L is a 8-bit processor with 8 kBytes of RAM 128 kBytes of FLASH ROM (program memory), and 4 kBytes of EEPROM (as mostly read-only memory). The microcontroller operates at about 3 V and draws about 8 mA and 8 μA in the active and the sleep modes respectively.

- **RF transceiver.** The CC1000 RF transceiver used in Cricket is configured to operate at 433 MHz [16]. This transceiver uses Manchester encoding and Frequency Modulation (FM) [57]. In Cricket, the transceiver is configured to send and receive data at a rate of 19.2 kilobits/s.

- **Ultrasonic transmitter.** The ultrasonic transmitter submodule drives a 40 kHz piezo-electric open-air ultrasonic transmitter at 12 V [102]. Under software control, this submodule generates ultrasonic pulses of duration 125 μs. The voltage multiplier module generates 12 V from the 3 V supply voltage to drive the ultrasonic transmitter.

- **Ultrasonic receiver.** We use an open-air type piezo-electric sensor that operates at 40 kHz to detect ultrasonic signals. The output of this sensor is connected to a two-stage amplifier with a programmable voltage gain between 70 dB and 78 dB. The ultrasonic signal is detected when the amplifier output goes above a preset threshold.

- **Expansion connector.** The Cricket node includes a connector that is pin-compatible with the Mica mote sensor interface. Mote-compatible sensor boards

can be connected to this board to build location-enhanced sensor networks. It is also possible to attach Mote processor boards to the Crickets for increased processing power. This connector can also be used to connect Cricket compass boards for obtaining listener orientation.

- **RS 232 interface.** An RS 232 interface with a DB-9 connector is used to attach a host device to the Cricket node.
- **Temperature sensor.** A pre-calibrated thermistor-type temperature sensor enables the beacons and the listeners to measure the ambient temperature to compensate for variations in the speed of sound with temperature.
- **Unique ID.** An 8-byte hardware ID, which is similar to an Ethernet MAC address, uniquely identifies every Cricket node.

3.3.1 Powering the Beacons and Listeners

Each Cricket node may be powered using two AA batteries, a power adapter, or solar cells.

Since beacons periodically transmit location information, they need a power source that can last for a sufficiently long time. A beacon consumes about 2.2 mA current on average and can operate on two AA batteries for five to six weeks. We have successfully powered the beacons using solar cells placed next to indoor lighting. A possible solution to reduce beacon power consumption is to use a low-power wake-up mechanism to wakeup the beacons when a nearby listener needs location information [89].

3.4 Chapter Summary

This chapter described the Cricket system architecture. This chapter also described the software and hardware architecture of Cricket.

The next chapter describes and evaluates the techniques for obtaining accurate beacon-to-listener distances in Cricket.

Chapter 4

Distance Estimation in Cricket

Cricket listeners use distance measurements to nearby beacons to compute listener location. Since typical indoor applications require accurate location information, listeners must be able to measure these distances accurately. Since beacon transmissions in Cricket are not centrally coordinated, the distance measurement technique also needs deal with possible interference among multiple beacon transmissions This chapter describes the techniques, algorithms, and the hardware and software implementations used in Cricket to obtain accurate beacon to listener distances. Section 4.1 describes the use of time-difference-of-arrival (TDOA) of RF and ultrasonic (US) signal to compute distances. This section also describes how various environmental factors affect the distance measurements, and examines Cricket's distance measurement accuracy.

The lack of centralized beacon coordination causes transmissions from different beacons to interact at the listener, resulting in incorrect distance samples. Section 4.2 describes how Cricket handles these incorrect samples using a combination of three algorithms: interference avoidance at the beacons, interference detection at the listener, and history-based filtering. Section 4.3 examines the throughput of a network of Cricket Beacons, in terms of both beacon transmission rate and the distance sample rate at the listener, using simulations and real-world experiments. We observe that periodic RF transmissions from a large number of deployed beacons cause scalability issues. Section 4.4 examines the scalability of the Cricket system by measuring the rate of RF collisions due to imperfect RF carrier sensing and by studying the immunity of the RF listeners to the interference from weak transmissions from far-off beacons. Finally, we describe the hardware implementation of Cricket beacons and listeners, and discuss various system deployment issues that arise in practice.

4.1 Distance Measurement from TDOA of RF and Ultrasound

Cricket uses the TDOA of RF and US signals to measure beacon-to-listener distances. Cricket beacons periodically transmit an RF message containing beacon specific information such as a unique identifier, coordinates, space ID, and measured ambient temperature. At the start of each message, each beacon transmits a narrow ultrasonic

pulse.

Since the velocity of RF is much larger than the velocity of sound, the US signal lags behind the RF signal as the two signals propagate. When a listener receives an RF message from some beacon, followed by an US signal, it measures the time interval δT between the start of the RF message and the arrival of the US signal at the listener. The listener can compute the distance d to the beacon from:

$$\delta T = \frac{d}{v_{\text{us}}} - \frac{d}{v_{\text{rf}}}.$$

At normal room temperature and humidity, the speed of sound, $v_{\text{us}} \simeq 344$ m/s, and speed of light, $v_{\text{rf}} \simeq 3 \times 10^8$ m/s. Since $v_{RF} \gg v_{US}$,

$$d \simeq \delta T \cdot v_{\text{us}}$$

To reduce power consumption and to keep the hardware simple, the US signal is only a narrow pulse that does not carry any data.

Because RF receivers take varying amounts of time to detect a valid preamble that precedes the actual RF message, the ultrasonic signal is transmitted at the start of the RF *message* rather than at the start of the RF *signal* itself; the receiver computes the time interval δT between the start of the RF message and the detection of the ultrasonic signal. The accuracy of the distance measurement depends both on the accuracy of measuring the time interval δT, and the accuracy of the estimated speed of sound used for distance computation. The accuracy of δT in turn depends on system parameters such as the RF bit rate, ultrasonic sensors and the detection circuits, and the timer resolution, while the speed of sound depends on environmental factors such as ambient temperature, humidity, and atmospheric pressure.

4.1.1 Environmental Effects on the Speed of Sound

The speed of sound in air depends on environmental factors such as temperature and humidity. In completely dry air with no humidity, the speed of sound (in meters per second) depends only on the absolute temperature T (in Kelvin), and is given by [58]

$$v_{US} = 20.05\sqrt{T}.$$

However, indoor air contains varying amounts of water vapor; with non-zero humidity, the speed of sound depends on the temperature, the relative humidity, and the atmospheric pressure. Lord, using work by Cramer and Davis, presents a more accurate equation for calculating speed of sound under varying environmental conditions [94, 23, 27]. The speed of sound is not very sensitive to relative humidity and atmospheric pressure variations. For instance, at 25 ^{0}C and 101.325 kPa (atmospheric pressure at sea level), the speed of sound changes by only about 0.5% as the relative humidity changes from 0% to 100%. At 25 ^{0}C and 50% relative humidity, the speed of sound changes by only about 0.6% as the atmospheric pressure changes from 101.325 kPa to 30 kPa (the atmospheric pressure at the top of Mount Everest). In contrast,

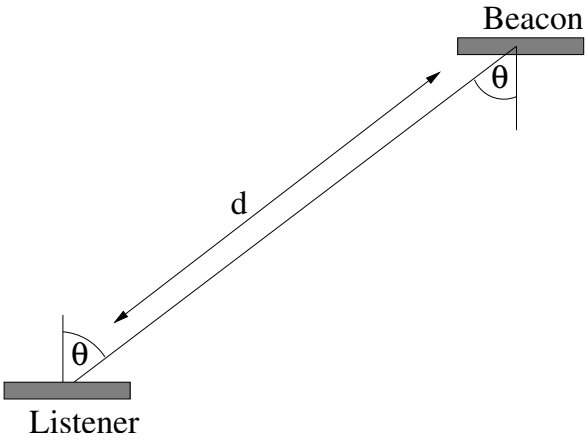


Figure 4-1: Experimental setup to determine the Cricket distance measurement performance.

the speed of sound changes by $\simeq 0.18\%$ per ^{0}C at 25 ^{0}C. Since the speed of sound has a relatively large sensitivity to temperature variations, and since indoor temperature can easily vary by even 10 ^{0}C within the same room, we use temperature sensors on Cricket beacons and listeners to compensate for changes in speed of sound due to temperature variations.

Each Cricket beacon measures the ambient temperature using an on-board temperature sensor, and includes this temperature in its RF message. When a listener L computes its distance to a beacon B, the listener measures its temperature t_L, and uses the value $(t_L + t_B)/2$ to represent the room temperature and computes the corresponding speed of sound. The cricket nodes can measure the ambient temperature with an accuracy of $\pm 1^0$C.

4.1.2 Distance Measurement Performance

We used the following experimental setup to measure Cricket's distance measurement accuracy. We deployed a transmitter and a receiver as shown in Figure 4-1; this setup mimics a beacon mounted on the ceiling, and a listener held parallel to the ground. Figures 4-2 and 4-3 show the error between the measured distance and the true distance for different values of d and θ. Each data point on the graph represents the mean absolute error, calculated over 100 samples; the vertical bars represent the minimum and maximum absolute error within the 100 samples. Because the ultrasonic sensors are not omni-directional, we could not get distance measurements for (d, θ) combinations that do not have a corresponding data point. We performed the experiment in a controlled environment to prevent outlier distance measurements due to reflected ultrasonic signals.

We observe that the absolute measurement error increases with the beacon to listener distance d. This increase is to be expected since increasing d causes the received US signal strength at the receiver to drop, causing the detection circuits to take a longer time to detect the signal, resulting in an increased positive error. The ultrasonic transmitter and receiver radiation pattern shown in Figure 4-4 (from

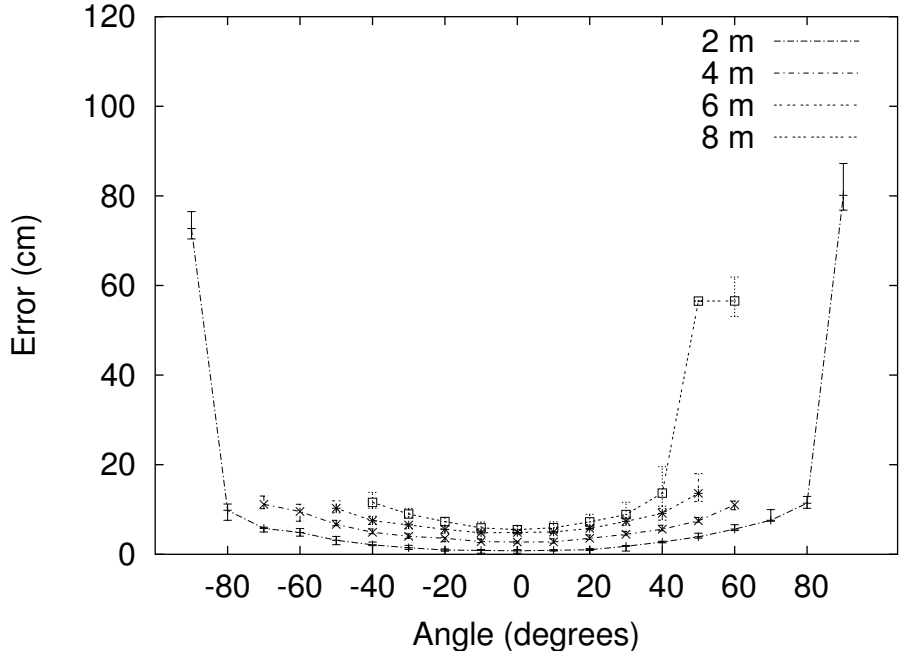


Figure 4-2: The distance measurement error as a function of the angle of rotation (θ in Figure 4-1) of the transmitter and the receiver at different transmitter to receiver distances.

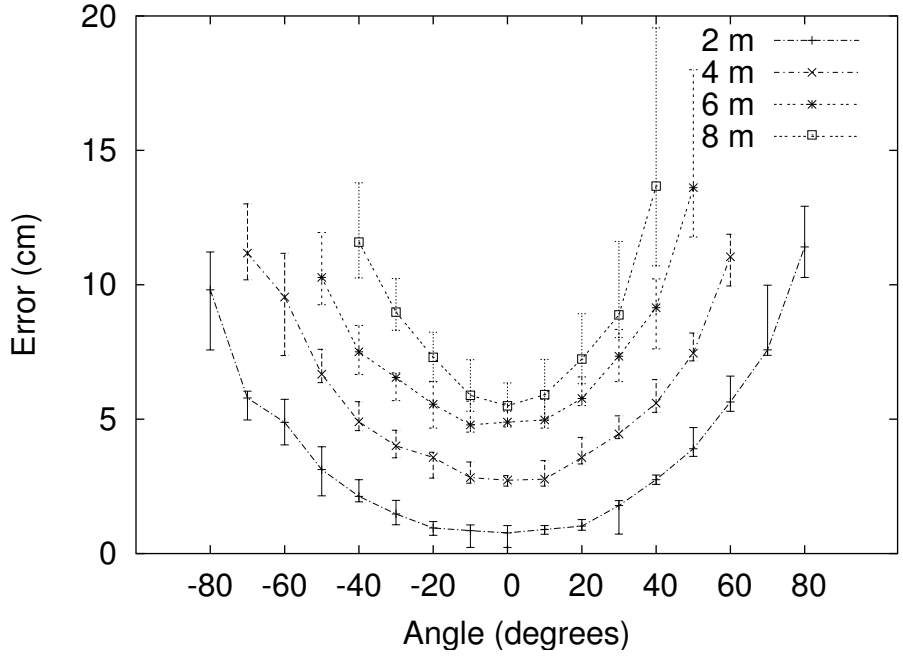


Figure 4-3: Zoomed-in version of Figure 4-2, showing the distance measurement error as a function of the angle of rotation (θ in Figure 4-1) of the transmitter and the receiver at different transmitter to receiver distances.

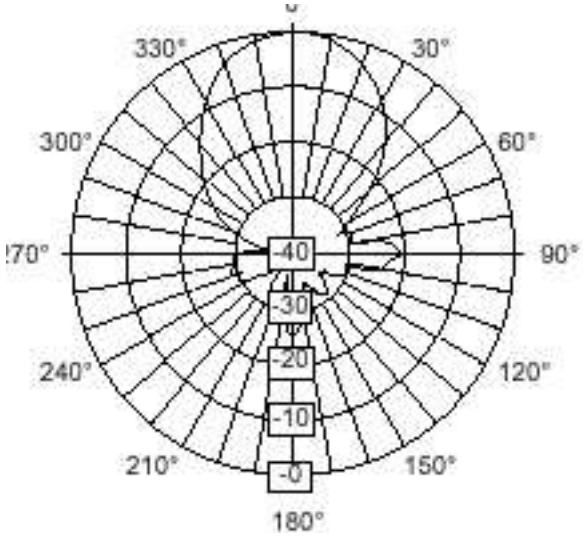


Figure 4-4: The radiation pattern of the Cricket ultrasonic transducer on a plane along its axis in (r, θ) polar coordinates. The r represents the signal strength (sensitivity) in dB and the θ represents the offset from the Z axis of the transducer.

the manufacturer's specification) explains the increase in error with increasing θ. As the radiation pattern shows, the transmitter and receiver sensitivity drops along directions that are away from the direction facing the ultrasonic transducer, hence with increasing θ. With increasing θ, the received signal strength at the listener decreases, again resulting in increased error. We observe that Cricket has a ranging accuracy of about 0.5% when the beacon and the listener are 2 m apart and are facing each other; however, the ranging performance degrades as we increase the separation and when they do not face each other. For θ in the range $(-40^o, 40^o)$, the error is under 5 cm. We also observe that for large angles, for example $\theta = 90^o$, there are no data points, because the ultrasonic signal at the listener is too weak to detect at large angles. A listener cannot measure distance to a coplanar beacon, when both the beacon and the listener are facing away from the plane, since this represents a θ of 90^o. Coplanar beacons, for instance when all the beacons are attached to the ceiling of a room, cannot measure inter-beacon distances although they can both transmit and receive RF and US signals.

4.1.3 Sources of Error in Distance Measurement

This section describes the possible sources of error when measuring distance between a beacon and a listener.

- *Environmental factors.* As we described in Section 4.1.1, dependency of velocity of sound on environmental factors such as temperature, humidity, and atmospheric pressure causes errors in velocity of sound-based distance measurements. Although it is possible to reduce these effects by measuring these environmental factors and compensating for them, it is not possible to completely eliminate

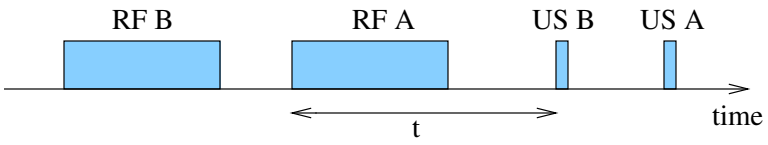


Figure 4-5: Inaccurate distance estimate caused by a listener using RF (RF_A) and ultrasonic (US_B) messages from different beacons to compute a distance estimate.

these effects since these factors can have different values along the path the sound travels.

- *Lack of line-of-sight.* If there is no line-of-sight path between the beacon and the listener, the sound may reach the listener after bending over an edge (refraction) or after reflecting off of some object. Both refraction and reflection cause the sound to travel a longer distance than the Euclidean distance between the beacon and the listener, resulting in distance measurement errors.

- *Errors in detecting US.* Cricket uses a threshold based approach to detect the arrival of the US signal. Cricket listener detects the arrival of a US signal when the signal amplitude at the output of the US amplifier circuit reaches a preset threshold (65 mV). However, the time taken for the received signal to reach this threshold is dependent on the received signal strength (as evident from Section 4.1.2). Hence, there is a received US signal strength dependent error in the distance measurement.

- *Timing quantization.* In TDOA based distance measurement, a measured time interval is converted in to a corresponding distance. This time interval measurement involves two types of quantization errors. First, measurement of time has a quantization error equal to the period ($\simeq$ 1 μs in Crickets) of the clock used for timing. Second, TDOA based approaches first detects the time of arrival of some reference signal, usually RF, to start the time interval measurement; detecting the arrival the RF message has a quantization error equal to the RF bit duration ($\simeq$ 52 μs in Cricket).

- *Variable interrupt service routine delays.* As described previously, RF and US based distance estimation requires the detection of RF arrival time. In Cricket, this detection is done within an Interrupt Service Routine (ISR) that handles the arrival of data form the RF transceiver. The invocation of this ISR can have a variable delay due to the presence of competing ISRs. This results in an error in the TDOA measurement.

- *Arithmetic quantization.* There can be quantization errors in the arithmetic routines that convert TDOA measurements to corresponding distances.

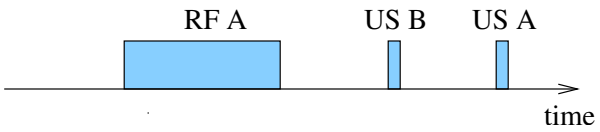


Figure 4-6: In US interference, a foreign US signal (US_B) arrives between the start of the RF (RF_A) and US (US_A) signals from some beacon A at a listener.

Figure 4-7: In RF interference, a foreign RF signal (RF_B) arrives between the start of the RF (RF_A) and US (US_A) signals from some beacon A at a listener.

4.2 Preventing Beacon Interference

While Cricket has the attractive property that the collection of decentralized beacons is easy to configure and manage, the absence of explicit coordination of beacon transmissions can cause signals from different beacons to interfere at a listener, resulting in incorrect distance measurements. Consider the RF signals RF_A and RF_B, and the US signals, US_A and US_B, of two beacons A and B received at a listener L. The signals RF_A and RF_B carry beacon specific data that enable L to identify their origins. In contrast, since the US signals in Cricket do not carry any data, L cannot differentiate between the signals US_A and US_B. With no coordination among beacons, the signals from A and B can interfere at L as shown in Figure 4-5. Here US_B arrives immediately after RF_A, but before the arrival of US_A. Since L cannot differentiate between US_A and US_B, it would use the time difference t between RF_A and US_B to calculate the distance to A, resulting in an incorrect distance sample.

Before we discuss how to prevent these incorrect samples, we identify two possible types of interferences. A foreign RF signal arriving between the RF and US signals from a beacon A, as shown in Figure 4-7, is called an *RF interference*, while a foreign US signal arriving between these signals (Figure 4-6) is called *US interference*. Combinations of these two types of interference are also possible.

We use a combination of three techniques to minimize, detect, and filter out these incorrect distances caused by foreign signals in Cricket. We first set the system's parameters to ensure that the RF range between a beacon and a listener is always greater than twice the US range. Under this assumption, we use a beacon scheduling algorithm that minimizes beacon interference, and an interference detection algorithm at the listeners to detect remaining interference. However, since we cannot make hard guarantees about indoor RF propagation, there can be instances where our assumption on the RF and US ranges fail. To deal with these situations, we use filtering algorithms that examines the history of distance samples to filter out the remaining incorrect distance samples.

It must be noted that interference avoidance in Cricket is different from interfer-

ence avoidance in traditional communication systems such as wired Ethernet, since Cricket uses two signals with different propagation characteristics, while traditional interference avoidance deals with only one signal. Before we discuss the interference avoidance algorithms in detail, we examine the salient characteristics of indoor RF and US signal propagation.

4.2.1 Indoor RF Propagation Characteristics

Cricket uses 433 MHz RF signals for transmitting location information. In free space, RF signals attenuate with the distance; since the RF signal strength at a receiver should be larger than some threshold for successful reception, there is some RF "range" beyond which a given transmitter and a receiver cannot communicate. In-building RF propagation is a well-researched topic, with most research concluding that, for indoor environments, RF signal strength and RF connectivity has no well defined correlation with the distance between the RF transmitter and the receiver. This lack of correlation is mostly due to the reflection and attenuation of RF signals by metallic objects [76].

Indoor RF signal propagation is subject to attenuation similar to the free space attenuation of RF signals, although the exact attenuation profile is not predictable. In the absence of wave guides (which are specially designed metallic tubes that enables RF signals to travel longer distances by repeated reflections), in-building RF range is typically less than the free-space range due to attenuation by various metallic and non-metallic obstacles. Apart from attenuation with distance, there are also local variations of RF signal strength due to destructive and constructive interference of signals reflected off of metallic objects (RF multipath effects). In general, we can provide almost 100% RF coverage inside a room by using an RF transmitter with high enough transmit power. However, as we increase the RF transmit power to achieve high coverage within a room, there will be significant RF leakage outside the room as well.

4.2.2 Indoor US Propagation Characteristics

US signals, at the frequency used by Cricket (40 kHz), do not penetrate physical objects such as walls. Since the wavelength of the US signals is smaller than the wavelength of the human audible sound, the ultrasonic signals do not diffract at openings such as doorways as readily as audible sound. Hence, the ultrasonic signals stay mostly confined inside an enclosed areas such as a room with only a limited leakage through openings such as doorways.

Figure 4-8 shows the physical construction of the US transmitters and receivers used in Cricket [102]. Since the construction of these transmitters and receivers is symmetrical about the Z axis, we assume that these transmitters and receivers have an omni-directional radiation pattern on the $X - Y$ plane. Figure 4-4 shows a (ρ, θ) polar plot of the radiation pattern of these transducers on a plane perpendicular $X - Y$ plane, going through the Z axis. Here, the r value represents the the signal strength (sensitivity) in dB and θ represents the angle offset from the Z direction.

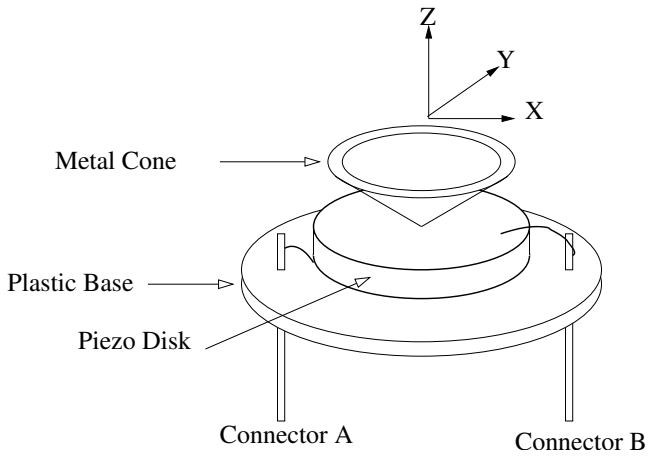


Figure 4-8: The internal construction of the ultrasonic transmitters and receivers used in Cricket. The two connectors A and B are attached to the top and the bottom of the piezo electric disk. The thickness of the piezo electric disk depends on the polarity and the magnitude of the voltage applied across these connectors. The dimensions of the piezo material causes the disk to resonate at 40kHz.

The figure shows that these transmitters and receivers have the highest sensitivity in the direction of the Z axis and the sensitivity drops as we move away from the Z axis. For example, the sensitivity drops to 1% (-20 dB) of the maximum at $\pm 50^o$ away from the Z direction. When an ultrasonic transmitter is attached to the ceiling of a room with the transmitter facing down, the received signal strength is a maximum when a receiver is located directly under the transmitter ($\theta = 0$). As the receiver moves away from the transmitter in a lateral direction, the received signal strength drops due to the combined effects of increased transmitter-receiver distance and due to the radiation pattern. This decay of ultrasonic signal strength results in a well-defined coverage area for a ceiling-mounted transmitter. Similar to RF, the size of this coverage area depends on the transmit signal strength and the receiver sensitivity. Since ultrasonic signals are easily reflected off of hard surfaces such as walls, a listener could receive a given ultrasonic transmission from multiple paths. Consider an ultrasonic transmission reaching a receiver along two paths: the direct path from the transmitter to the receiver (P_1), and a path where the signal gets reflected off an obstacle such as a wall (P_2). Let these path lengths be l_1 and l_2 respectively. For a given transmission, the time gap δt between the arrival of signals along these paths at the receiver is given by:

$$\delta t = \frac{l_2 - l_1}{v_{\text{us}}}$$

In an indoor environment, the value $l_2 - l_1$ can be several meters, causing δt to be several milliseconds long. For example, if $l_2 - l_1 = 2$ m, $\delta t \simeq 6$ ms. Because of reflections, a single transmitted wavefront causes a series of received signals with different delays, usually resulting in a wide pulse—with a duration of several milliseconds—at

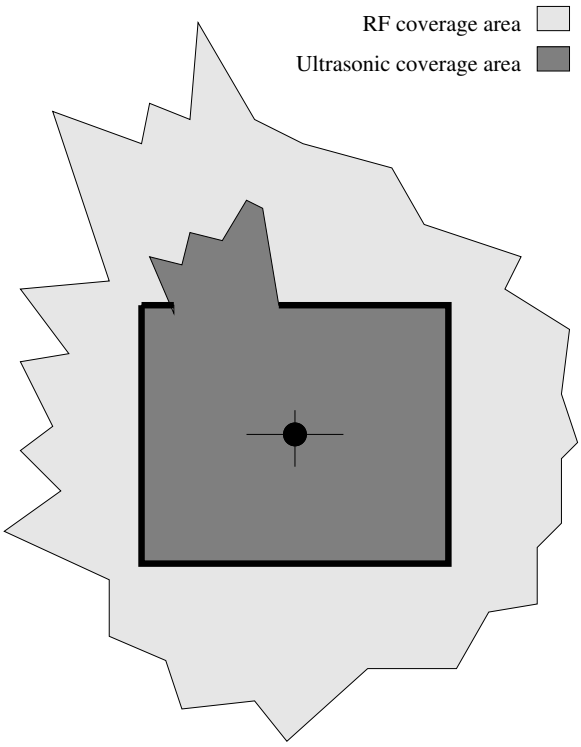


Figure 4-9: Although indoor propagation of RF and US is not predictable, we can ensure that RF range $> 2 \times$ US range most of the time by increasing the RF transmission power as necessary. The inability of US signals to permeate obstacles also tends to reinforce this relationship.

the receiver.

At a speed of v_{us}, the ultrasonic signal travels a distance d within a time d/v_{US}. If the ultrasonic transmitter and the receiver have a maximum range of R_{us}, the ultrasonic signal can travel for time at most $R_{\text{us}}/v_{\text{us}}$. If the duration (width) of the ultrasonic transmission is t_{us}, the ultrasonic signal must "disappear" within time $D_{\text{us}} = R_{\text{us}}/v_{\text{us}} + t_{\text{us}}$. With $R_{\text{us}} \simeq 10$ m and $t_{\text{us}} = 250$ μs, an ultrasonic transmission from a Cricket beacon completely dies down after about 30 ms.

4.2.3 Interference Avoidance Algorithm at the Beacons

Since both RF and US transmissions have a limited range, and since we can increase the RF range by increasing the RF transmission power as necessary, we can set the RF transmission power such that RF range, R_{rf} and US range R_{us} satisfy the condition $R_{\text{rf}} > 2 \times R_{\text{us}}$ with high likelihood (we cannot guarantee a probability of 1 due to localized variations of RF signal strength arising from multipath effects). As Figure 4-9 shows, the inability of US signals to penetrate obstacles reinforces this relationship.

We observed that the US signals from a given beacon transmission disappear within an interval D_{US}; the duration of the beacon RF message is made smaller than

```
while true do
    r ← randomUniform(T1,T2);
    delay(r);
    startRF_Rx();
    delay(D_us);
    if no_beacon_message and RF_carrier_free then
        transmitBeacon(RF,US);
    endif
endw
```

Figure 4-10: Beacon transmission scheduling algorithm.

D_{us}. Since both RF and US signals from a beacon transmission cease to exist after D_{us}, the signals from two beacon transmissions that are separated by more than D_{us} cannot interfere at a listener.

We use the beacon transmission scheduling algorithm shown in Figure 4-10 to minimize interference among beacons that are within RF range, by separating their transmissions by at least D_{us}. Each beacon sleeps for some random interval r; next, before transmitting an RF signal, the beacon turns on its RF receiver and waits for D_{us} to see if it hears a beacon message within this interval; if it does not hear any beacon message within D_{us}, the beacon transmits its own beacon message after RF carrier sensing to ensure that there are no ongoing beacon transmissions. It must be noted that, during the D_{us} period, the beacon specifically checks for beacon messages but ignores other message types.

This algorithm ensures that, under perfect carrier sensing (where a beacon can detect all the ongoing RF transmissions from other beacons within its range before starting its own transmission), RF messages from two beacons within each others RF range are separated by at least D_{US}. The relationship $Range_{RF} > 2 \times Range_{US}$, guarantees that if some listener is within the US range of two beacons, then those two beacons must be within each other's RF range. Since transmissions from beacons within each other's RF range are separated by at least D_{us}, this scheduling algorithm prevents US interference among beacons.

However, this algorithm prevents only a subset of possible RF interference. For example, in Figure 4-11, the listener is within the RF range of the two beacons while the two beacons are not within each other's range. Hence, the RF transmissions from these two beacons can interfere at the listener. This is the well known *hidden terminal effect* in RF communication. We use interference detection algorithms at the listener to detect these RF interferences due to hidden terminal effect.

Beacon Interference Avoidance Experimental Results

We used the following experiment to evaluate the performance of the beacon interaction avoidance algorithm. As shown in Figure 4-12,we deployed 50 beacons within a 2 m $\times$ 2 m area; and a listener 2.5 m away from the plane containing the beacons. We observed the distance estimates at the listener as we turned on the beacons in groups of five. Distance measurements at the receiver that deviated by more than 4 cm from

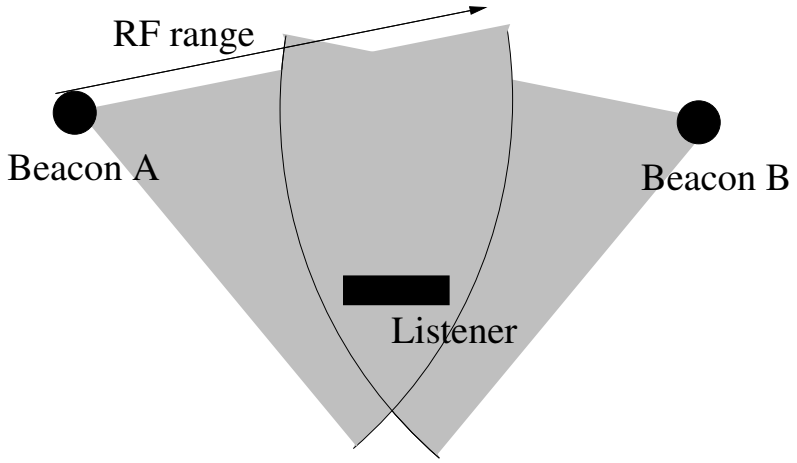


Figure 4-11: The listener (receiver) is within the RF range of the two beacons (transmitters). But the two beacons (transmitters) are not within each other's RF range. This is called the *Hidden Terminal Effect*.

the true distance were treated as incorrect distance samples. Figure 4-13 shows the percentage of outlier distance estimates at the listener as we vary the number of active beacons. We observe that the interference avoidance algorithm works well since even 50 in-range beacons result in less than 1.7% of incorrect distances.

4.2.4 Interference Detection at the Listener

As we describe above, due to the hidden terminal effect, the beacon scheduling algorithm does not prevent all possible RF interference among beacons. Cricket listeners use an interference detection algorithm to detect and remove incorrect samples caused by RF interference. Consider a listener using an RF and US signal combination to measure distance to a beacon. We examine the number of RF messages that the listener receives during the time interval D_{us} before the start of the US signal.

We first assume that there are no overlapping RF messages during this interval. Under this assumption, if the listener has received exactly one RF message during D_{us}, it is almost certain that the RF and US signals came from the same beacon, and the listener uses these two signals to compute the distance to the corresponding beacon. If the listener received more than one RF message during D_{us}, then the listener cannot be certain about which RF message the US signal corresponds, and the listener discards the US signal and the RF messages.

Next we relax our assumptions and examine the effects of possible RF message overlap during the D_{us} interval. When two RF messages, RF_A and RF_B from two beacons A and B overlap at a listener, there can be three possible outcomes: the listener receives the message RF_A, or the listener receives the message RF_B, or the listener receives neither message due to the bit errors (and the corresponding CRC failures) caused by overlapping RF messages. If one of the overlapping RF messages, say RF_A, came from the same beacon as the US signal under consideration, it is important for the listener to receive the message RF_A correctly to prevent an incorrect

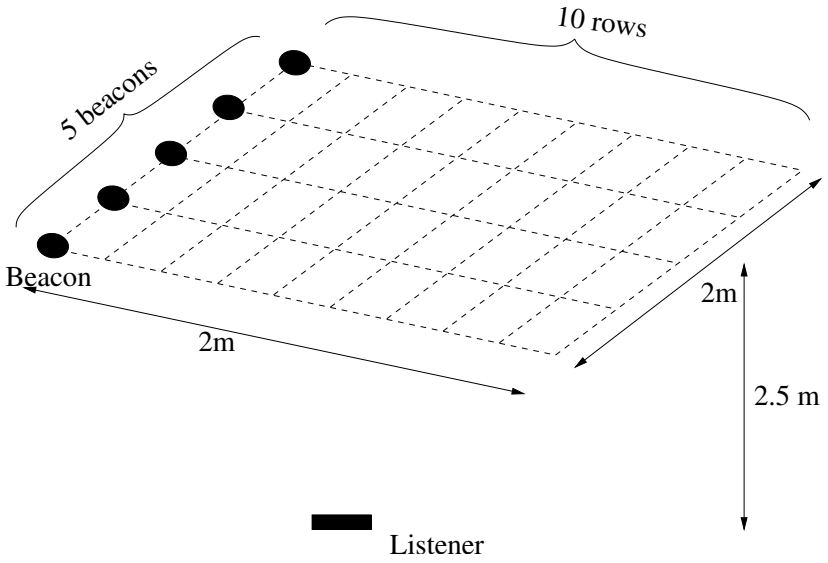


Figure 4-12: The beacon and listener deployment to determine Cricket collision avoidance performance. We deployed 50 beacons in increments of 5.

distance sample. Since the two RF messages are separated by less than D_{us}, under the assumption of perfect carrier sensing, the distance between the two beacons A and B must be greater than the RF range. Since the US signal at the listener came from A, and $Range_{RF} > 2 \times Range_{US}$, the listener must be closer to beacon A than beacon B and there must be a line-of-sight path between beacon A and the listener. Because of these reasons, it is likely that RF_A signal is stronger than RF_B signal. According to the experimental results presented in section 4.4, the RF receiver in Cricket can correctly receive the stronger message when two RF messages overlap; hence, when RF_A and RF_B overlap, it is likely that the listener will correctly receive the message RF_A. We conclude that the listener interference detection algorithm works even when the interfering RF messages overlap.

Figure 4-14 shows the pseudocode of the listener interference detection algorithm. Each listener keeps track of the timing of the two latest RF ranging messages it has received. When the listener receives an ultrasonic pulse, it checks if it has received more than one RF ranging message during the time interval D_{US} before the arrival of the ultrasonic signal. If it has received more than one RF ranging message, then it discards the messages and the US signal.

4.2.5 Algorithms for Filtering Incorrect Distance Samples

The interference avoidance and detection algorithms described in the previous sections prevent an incorrect distance sample from being inferred at the listener, if certain assumptions on RF and US propagation hold. In practice, RF dead spots due to destructive multipath interference and imperfect carrier sensing result in a few incorrect distance samples at the listeners. Since the beacon transmission algorithm in Figure 4-10 uses a random interval between consecutive transmissions, repeated

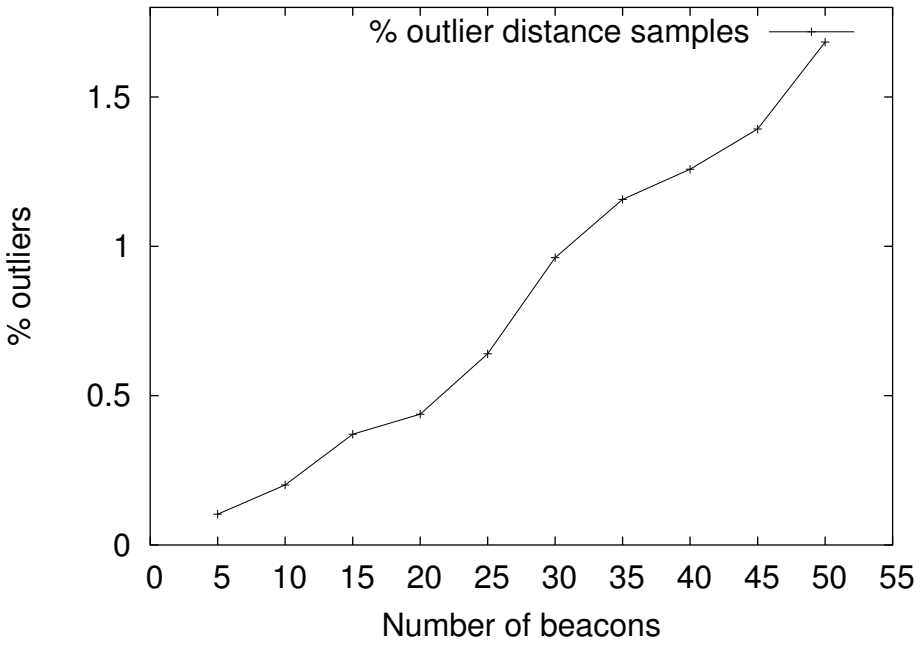


Figure 4-13: Percentage of outlier distance samples at a listener as a function of the number of in-range beacons. An outlier is a distance sample that was more than 4 cm off from the true distance.

interference between a pair of beacons has a low likelihood. We use this property to develop filtering algorithms that examine the history of distance samples to filter out the incorrect samples. The MinMode algorithm described below successfully filters out incorrect samples when the receiver is stationary or is moving slowly.

The MinMode algorithm first collects a set of distance samples from each near-by beacon. Then it rounds-off these values to compensate for small measurement errors and for slow movement of the receiver node. Next it selects the distance with the highest frequency of occurrence as the correct distance to the corresponding beacon. If there are multiple distances with the same frequency of occurrence, it selects the minimum of these distances. Since incorrect distances have a low probability of recurring, compared to the correct distance, this algorithm filters out the incorrect distances. However, for mobile listeners, the MinMode algorithm performance degrades with the increasing speed of the listener since at high speeds individual distance samples to a given beacon vary by a large amount. In Section 5.2.2 we describe a Kalman filter-based approach to filter out incorrect distances.

4.3 Cricket Throughput Performance Analysis

The position estimation accuracy of a mobile listener depends on the frequency of the distance samples the listener receives from nearby beacons; with a higher frequency resulting in a better accuracy. In this section we determine the Cricket beacon throughput, defined as the rate of distance samples at a listener from a given

```
handle_RF_rx( message){
    if message.type = RANGING_MESSAGE then
        first_message_time = second_message_time;
        second_message_time = current_time();
    endif
    ...
}

handle_ultrasonic(){
    if current_time() - first_message_time > T_US then
        //filter rf interactions
        report_distance(current_time() - second_message_time);
    endif
}
```

Figure 4-14: Timing-based interference detection at the listener.

beacon. In this section we examine the performance of beacon scheduling algorithm using both simulation and experiments. For simulations, we implemented a simple discrete event driven simulator.

4.3.1 Beacon Transmission Throughput

We use simulations to determine the throughput of the beacon scheduling algorithm given in Figure 4-10. To reflect the actual implementation, we use the following parameters in the simulations: `T1` = 650 ms, `T2` = 1350 ms, and D_{us} = 50 ms, and *beacon message duration* = 15 ms. We simulated the beacon scheduling algorithm for different numbers of in-range nodes n; for a given n, we ran the simulator for 1000 simulated seconds.

Figure 4-15 shows the rate of successful transmissions r per beacon as a function of n. We observe that the per node transmission rate drops rapidly with n.

Figure 4-16 shows the aggregate message transmission rate for n nodes as a function of n. When n is small, the transmission rate increases approximately linearly with n, since there is enough time between consecutive transmission attempts from a single beacon to accommodate almost all the transmission attempts from other beacons. As n grows, the rate of successful transmissions asymptotically reaches $1/D_{\text{us}}$, since the beacon transmissions start to occupy the whole channel. We observe that, with increasing n, the RF channel utilization improves, although the transmission rate per beacon drops.

4.3.2 Throughput at the Listener

This section examines the rate of successful distance samples at a listener from a beacon located within the US range of the listener as a function of the distance between the beacon and the listener and the beacon density. For a given beacon, the successful transmission rate of the beacon (Figure 4-15), and the rate at which

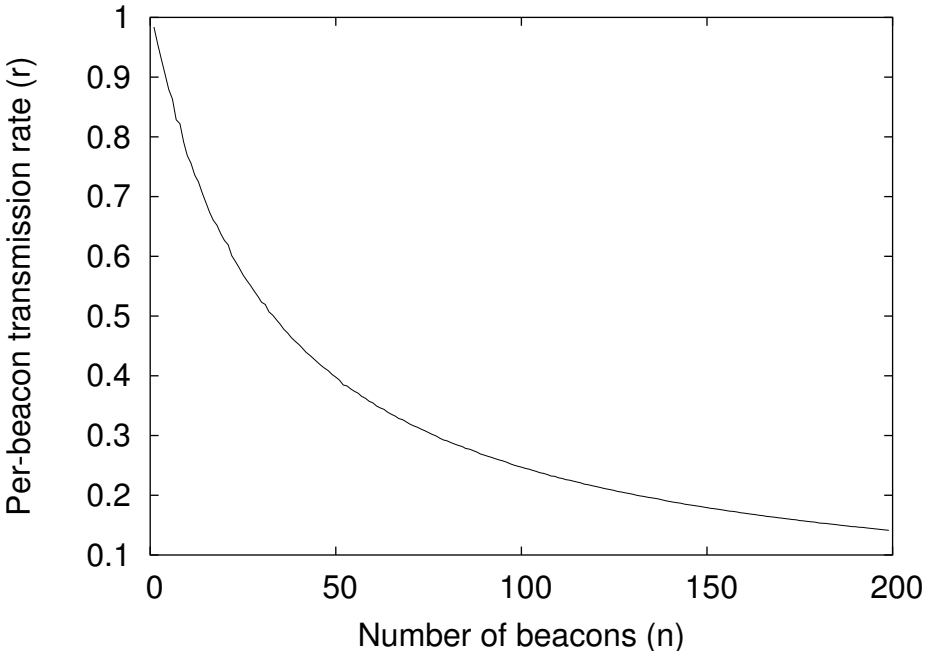


Figure 4-15: Simulation results of per beacon transmission rate as a function of the number of in-range beacons.

successful distance samples from that beacon are received at a given listener, can be different since the interference detection algorithm at the listener throws away interfering RF and US signals. The interference detection algorithm (Figure4-14) at the listener throws away RF and US signals when more than one RF message is received during the time interval D_{us} before a US signal. To simplify our analysis of successful distance sample rate at the listener, we assume that the listener interference detection algorithm discards RF transmissions (and associated US signals), that come from beacons that are within listener's RF range, which are separated by less than D_{us} .

Since transmission from beacons within each other's range are assumed to be separated by more than D_{us}, only transmissions from beacons not within each other's range can be separated by less than D_{us}. We also make the assumption that there exists a fixed distance R_{RF}, the "RF range", such that RF transmissions from some beacon A are received by all the beacons and listeners located within a distance R_{RF} from A; and, the RF transmissions from A do not have any impact on beacons and listeners that are located at a distance more than R_{RF} from A. Under this assumption, if the two beacons A and B are not located within R_{RF} from each other, but if both A and B are located within distance R_{RF} from some listener L, then all RF (and corresponding US) transmissions from A and B, which are separated by less than D_{us} are discarded by the interference detection algorithm of L.

Given a uniform beacon deployment, next we determine the number of beacons whose RF transmissions can interfere with the transmissions from a given B at a given listener L located at a distance r from B. Consider the beacon B and the

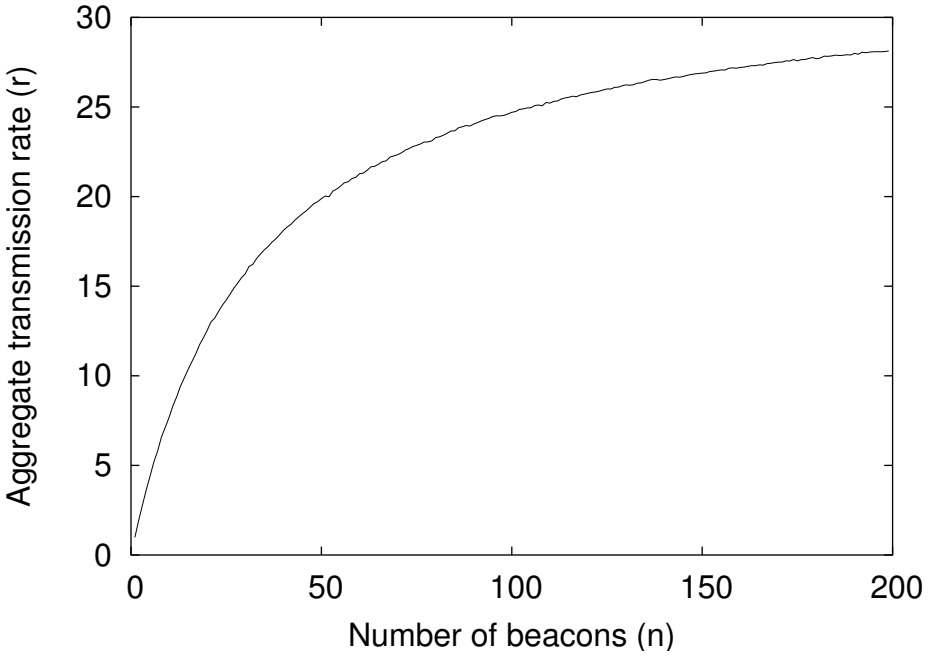


Figure 4-16: Simulation results of the aggregate beacon transmission rate as a function of the number of in-range beacons.

listener L in Figure 4-17, which are located on the same plane. Since the shaded area is not within distance R_{RF} from B, but is within R_{RF} from L, transmissions from B and the transmissions from the beacons in the shaded area can arrive within time D_{us} of each other at L. The area of the shaded area A_s is given by:

$$A_s = \pi {R_{RF}}^2 - 2A,$$

where,

$$A = \frac{{R_{RF}}^2}{2}(\theta - \sin\theta) \text{ and}$$

$$\theta = 2\arccos\left(\frac{r}{2d_c}\right).$$

If beacons are uniformly deployed, and if there are n beacons within a circle of radius R_{RF}, the number of beacons within the shaded area n_s is given by:

$$n_s = n\frac{A_s}{\pi {R_{RF}}^2}.$$

We used the following experiment to evaluate the rate of successful distance samples at a listener L from a beacon B. We simulated two groups, G_1 and G_2 of n beacons running the scheduling algorithm in Figure 4-10 for 10,000 simulated seconds, and logged all the beacon transmission times. Next, assuming all the n nodes are uniformly distributed, we obtained the the number of nodes, n_s, that can inter-

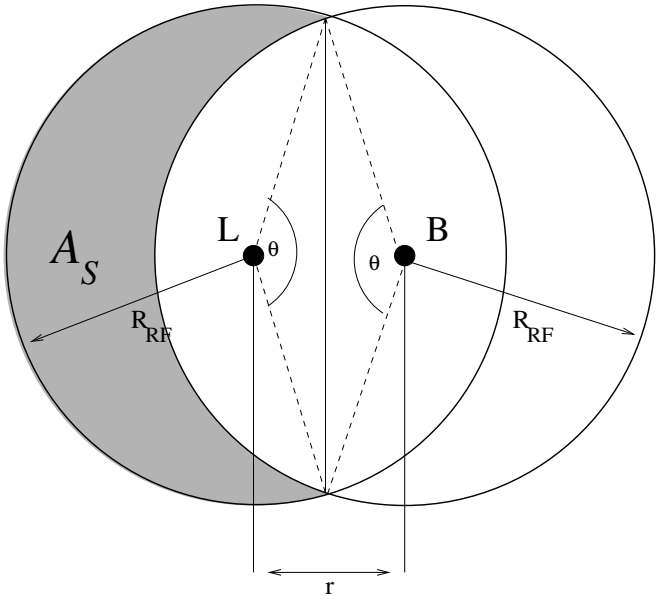


Figure 4-17: The shaded area shows the location of the subset of beacons whose transmissions can be received at listener L, separated by less than D_{US} from the transmissions of the beacon B.

fere at a listener located at a distance d from a beacon. We selected a beacon, b_0 from group G_1 and n_s beacons from group G_2. Next we used the transmission times to determine the fraction of transmissions from the beacon b_0 that are separated by more than D_{us} from the transmissions from the collection of n_s beacons; these are the successful transmissions that result in valid distance samples at the listener. Figure 4-18 plots the percentage of successful transmissions as a function of the separation d between the beacon and the listener.

4.3.3 Throughput Experimental Results

We used the following experiment to investigate the throughput of a Cricket beacon deployment. We deployed 50 beacons in a 2 m $\times$ 2 m area, and collected distance samples at a listener placed 2.5 m away from the beacons (Figure 4-12). Figures 4-19 and Figure 4-20 plot the aggregate and the per-beacon, successful distance sample arrival rates at the listener, respectively.

We observe that for $n = 10$ beacons, the sample rate in Figure 4-19 approximates the sample rate obtained from simulations (Figure 4-16). However, unlike simulations where the sampling rate increases with increasing n, the sample rate in the experiment drops as we increase n beyond 10. We attribute this behavior to the following. First, increasing n corresponds to an increased workload at the listener. The combination of increased US and RF interrupt processing, increased communication with the host and increased computation can lead to "livelock" at the listener [74]; although the microcontroller may be capable of handling the average workload, the peak workload may prevent it from servicing the periodic RF interrupts in a timely manner, leading

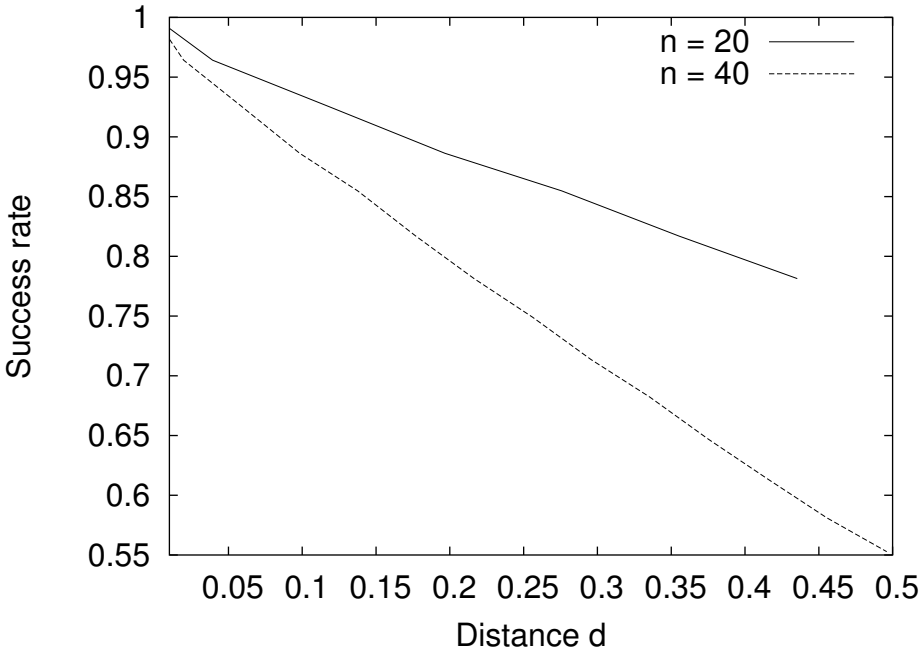


Figure 4-18: The percentage of distance samples that are successfully received at a listener as a function of the distance between the beacon and the listener (assuming all the nodes are located on the same plane).

to dropped packets. Next, the increased n may lead to increased RF interference. For example, the fraction of RF collisions due to imperfect carrier sensing increases with increasing n; with the dense beacon deployment used in the experiment, the antennas on beacons can act as shadows resulting in RF deadspots where a beacon may not be able to receive RF transmissions from a nearby beacon, resulting in increased RF collisions at the listener.

4.4 Cricket Scalability

As described in Chapter 5, a Cricket listener needs only three or four in-range beacons to obtain accurate location information. Since there is only limited ultrasonic interaction among beacon transmissions in an indoor deployment (section 4.2.2), Cricket's ultrasonic performance does not degrade significantly with an increasing number of beacons, as long as the beacon density is kept low (e.g., only three or four beacons visible from a given listener position). However, as we discussed in section 4.2.1, there is significant RF leakage beyond the coverage area of a given beacon, resulting in increased RF interaction among beacons in a large Cricket deployment. This section investigates the scalability of a Cricket deployment by studying the performance of the RF technologies used in Cricket.

Section 4.4.1 examines the carrier sensing performance, of the radios used in Cricket by measuring the time taken for the radio to transmit an RF message af-

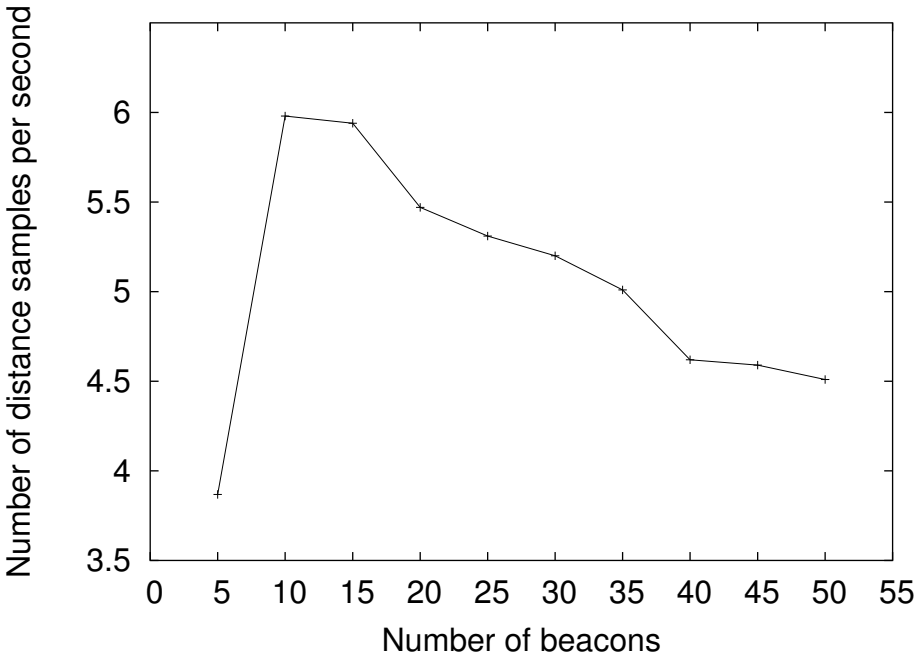


Figure 4-19: The distance sample rate at a listener as a function of the number of in-range beacons.

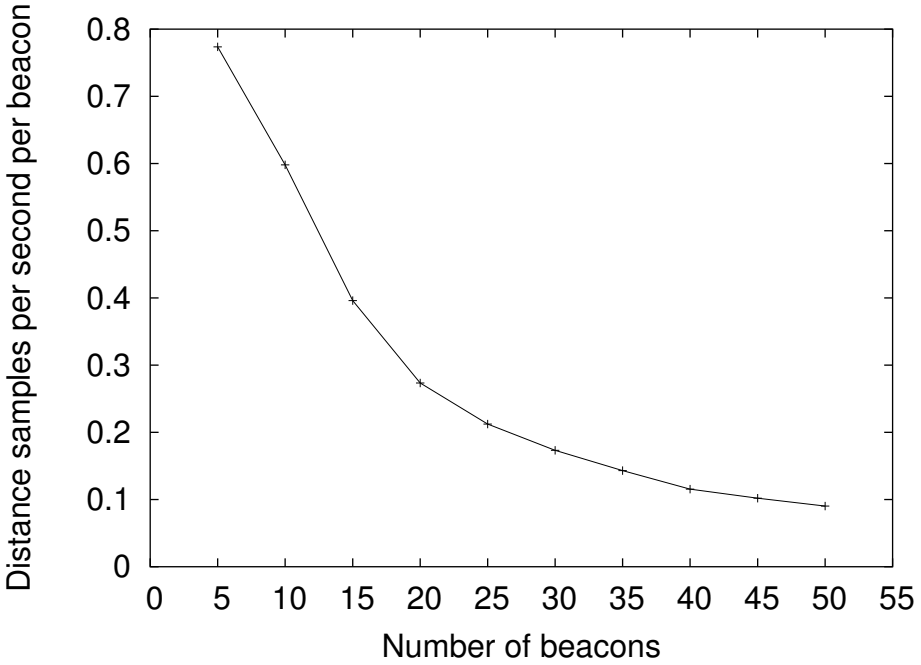


Figure 4-20: Per-beacon distance sample rate at the listener as a function of the number of in-range beacons.

ter detecting a free RF channel. This is an important parameter since this delay represents a vulnerability interval during which transmissions from two carrier sensing radios can collide, with a larger delay resulting in a higher probability of collisions. Section 4.4.2 examines the ability of the radio to successfully receive an RF message in the presence of overlapping weaker RF transmissions. This is an important measure of the ability of a Cricket listener to correctly receive RF messages from near-by beacons in the presence of weak RF transmissions from far-away beacons; ability to correctly receive strong RF messages in the presence of weak over lapping RF transmissions is also important for the correct operation of the listener interference detection algorithm (Section 4.2.4).

4.4.1 RF Carrier Sensing Performance

The beacon interference avoidance algorithm (Section 4.2.3) performs well assuming ideal RF carrier sensing among in-range beacons. In practical implementations of RF carrier sensing, a node turns on its radio receiver and checks if an RF carrier is present. If the carrier is free, the node transmits its RF data. Cricket beacons use an RF transceiver that can act as either a RF transmitter or a receiver for sending and receiving RF messages. The microcontroller on the beacon switches the transceiver between the transmit and the receive modes by writing to configuration registers in the transceiver. However, it takes a non-zero amount of time for the transceiver to make the transition from receive mode to transmit mode. This transition time includes the time taken to update the configuration registers and the time taken by various submodules within the transceiver to power up and settle. Hence, there is a finite amount of delay between a beacon detecting a free RF carrier and the start of RF signal transmissions. We denote this "carrier sense delay" by d_{CS}.

Consider a beacon A that decides to transmit an RF message upon detecting a free RF carrier. It will take d_{CS} for A to generate an RF signal from the instant that A decided that the RF channel was free. If some other beacon B checks the RF carrier within the interval d_{CS}, it will also detect a free RF carrier and will decide to transmit. Since d_{CS} is much smaller than the RF message length, the RF transmissions from A and B will collide. Similarly, if B had checked the RF carrier and had decided to transmit during the interval d_{CS} before A checked the RF carrier, A will still detect a free carrier and the transmissions from A and B will collide. Hence a non-zero d_{CS} can cause transmissions from beacons that can usually detect each other's RF signals to collide, resulting in incorrect distance samples at the listener.

We used the setup shown in Figure 4-21 to determine the value of d_{CS} for our implementation. The Cricket node labeled "Controller" periodically generates two pulses P_A and P_B on two digital I/O lines of the on-board microcontroller at a frequency of 1 Hz; P_A is generated first and P_B is generated after a delay δt. These two lines are connected to the hardware interrupt pins of the microcontrollers on the two Cricket nodes A and B. The radios on both A and B are in receive mode. The interrupt service routine in A switches the radio to transmit mode, and, after a delay of 50 ms, switches the radio back to receive mode. The interrupt service routine in B checks if the RF carrier is busy and reports this information to the attached

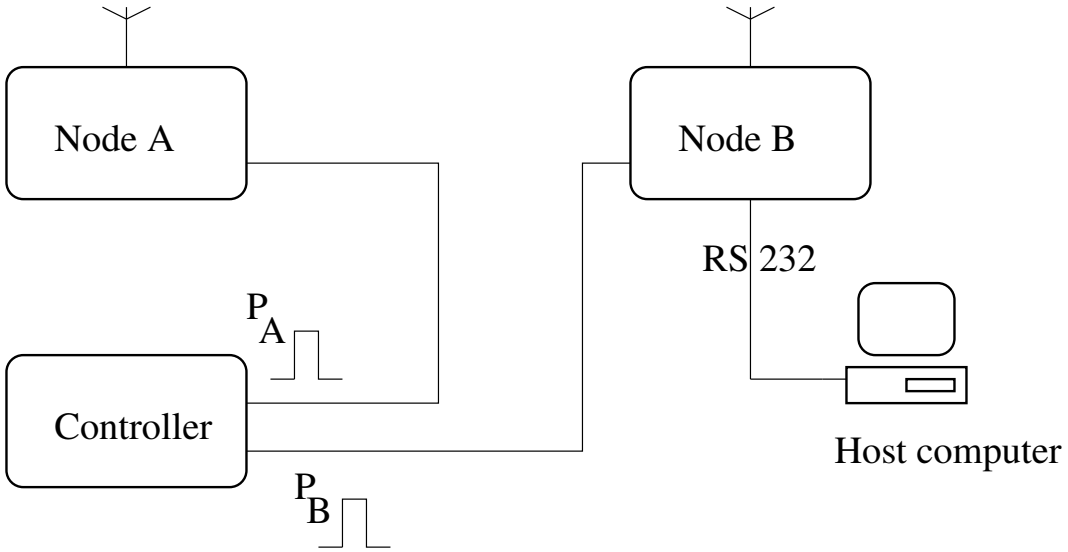


Figure 4-21: Experimental setup to determine the carrier sensing delay, d_{CS}. The controller generates two pulses P_A and P_B separated by a delay δt. When P_A arrives, node A starts its RF transmitter; when P_B arrives, node B checks the RF carrier status and reports it to the attached host, which logs this data.

host computer. We ran this experiment for different values of δt from 0 to 380 μs in increments of 10 μs; for each value of δt, we ran 100 instances of the experiment.

Figure 4-22 plots the percentage of time that node B detected an RF carrier, as a function of the delay δt. We observe a clear transition in carrier sensing performance at 320 μs. This transition indicates that it takes a delay of 320 μs for the RF signal from node A to appear since A started to switch from receive mode to transmit mode, this delay corresponds to the carrier sensing delay, d_{CS}. From this data, we conclude that d_{CS} for our system is 320 μs.

4.4.2 RF Collision Performance

In a large beacon deployment, apart from the strong RF transmissions from nearby beacons, a listener will receive a large number of weak RF transmissions from far-away beacons. At the listener, RF transmissions from far-away beacons will collide with transmissions from nearby beacons due to the hidden terminal effect. In this section, we investigate the impact of these RF collisions. This section investigates RF receiver performance when RF transmissions with different signal strengths collide at a Cricket listener.

We use the setup shown in Figure 4-23. Here, the controller generates three signals (pulses) P_A, P_B, and P_C. These signals are connected to interrupt pins of the microcontrollers on Cricket nodes A, B, and C. In nodes A and B, the interrupt service routine invoked by signals P_A and P_B transmits an RF message without carrier sensing. To clearly identify RF collisions, the RF messages transmitted by A and B do not contain any common data, except for the common RF preamble.

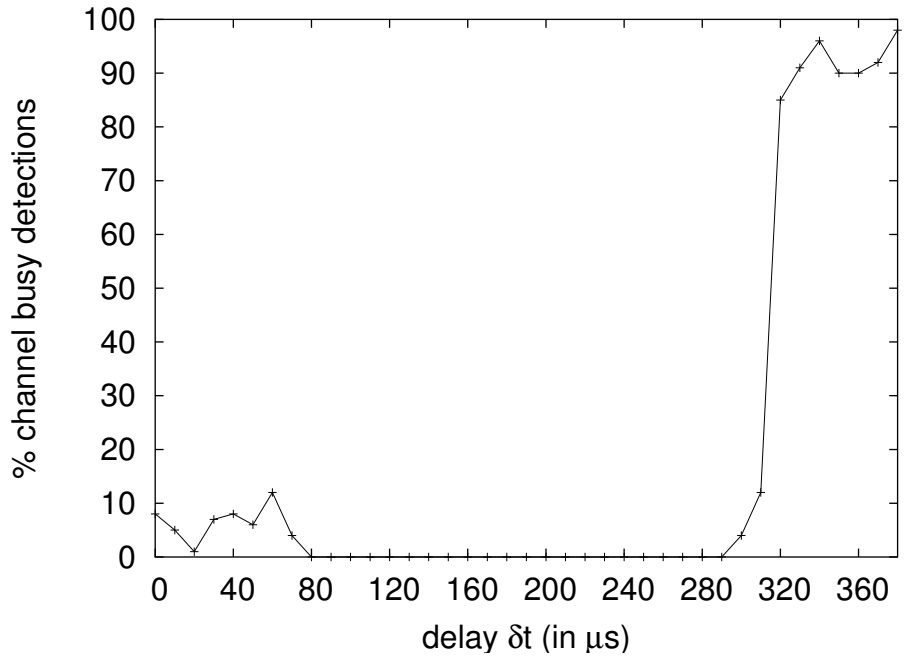


Figure 4-22: The percentage of time the RF carrier is detected by node B as a function of the delay between node A starting its RF transmitter and node B sensing for RF carrier.

The controller has four stages of operation. In stage 1, the controller generates 100 P_A pulses at a frequency of 1 Hz. In stage 2, it generates 100 P_B pulses at 1 Hz. In the 3rd stage, the controller generates 100 instances of P_A and P_B at 1 Hz, with both P_A and P_B starting at the same time; hence, in this stage A and B generates fully overlapping RF signals. In the 4th stage, the controller generates 100 instances of P_A followed by P_B with a time gap of 6 ms between the pulses; 6 ms corresponds to the half the duration of the RF signal, which causes the signals from A and B to partially overlap in the middle of the A's message. At the start of each stage, the controller generates the signal P_C so that node C can keep track of the current state of the controller.

During stage 1, node C computes and logs the average received signal strength of RF messages from A. In stage 2, C logs the signal strength of messages from B. In stages 3 and 4, C logs the following information: the number of RF messages c_A received from A, the number of messages c_B from B, and the number of messages c_E with bit errors (CRC errors). Since A and B transmit 100 instances of closely spaced messages, C can compute the number of losses, $c_L = 100 - (c_A + c_B + c_E)$. We ran different instances of this experiment by placing A, B, and C in different configurations, within a 4 m $\times$ 4 m area, to obtain different RF signal strengths at C.

Figure 4-24 shows the percentage values of C_A, C_B, C_E, and C_L for different values of $RF\_RX_{SS}(A)/RF\_RX_{SS}(B)$ for the 3rd stage of the experiment where A and B simultaneously transmits RF signals; here, $RF\_RX_{SS}(X)$ represents the signal strength

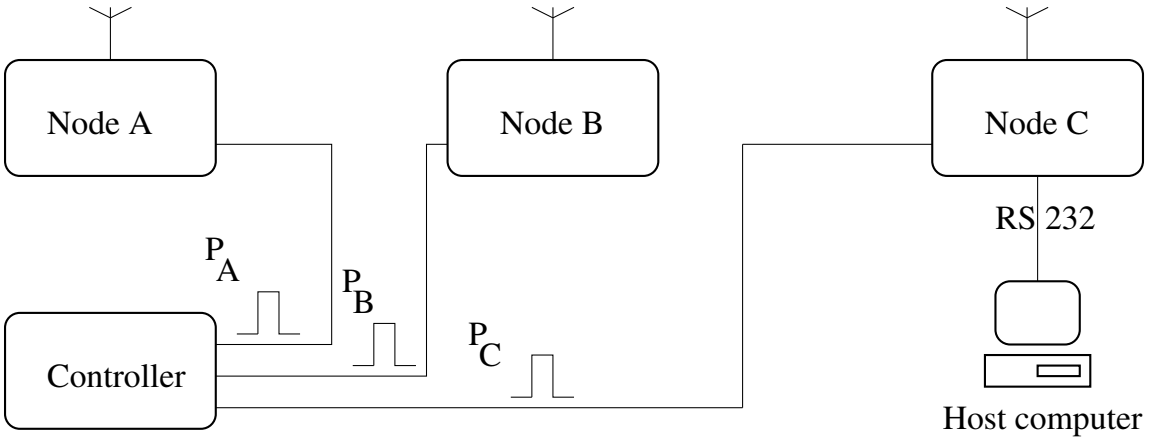


Figure 4-23: Experimental setup to determine the RF collision performance at a Cricket listener.

of RF transmissions from node X at the receiver C. We observe that the number of lost packets and packets with CRC errors increase when $RF_RX_{SS}(A)/RF_RXSS(B)$ approaches 1. This behavior is expected since two overlapping RF signals with equal strengths cause bit errors that prevent the receiver from correctly decoding either RF message. More importantly, we notice that when the signal strengths are only slightly different, the receiver can correctly decode a large fraction of packets from the node with stronger RF signal. For example, when $RF\_RX_{SS}(A)/RF\_RXSS(B) = 0.811$, where the signal strength of B is $\simeq 1.23$ time the signal strength of A, $\simeq 95\%$ of B's transmissions are successfully recived at C; when $RF\_RX_{SS}(A)/RF\_RXSS(B) = 1.24$, $\simeq 94\%$ of A's transmissions are successfully received at C. The Cricket radio can correctly decode the stronger RF message in the presence of weaker interfering RF signals that overlap with the stronger signal. This is called the *capture* effect where a strong signal completely dominates a weak signal at an RF receiver. The Frequency Modulation (FM) used in Cricket radios to encode data on to the RF signal has better capture performance compared to schemes such as Amplitude Modulation (AM).

Figure 4-25 shows the percentage values of C_A, C_B, C_E, and C_L for different values of $RF\_RX_{SS}(A)/RF\_RXSS(B)$, for the 4th stage of the experiment, where A transmits first and B transmits after a delay of one-half the RF message duration. We observe that when $RF\_RX_{SS}(A) > RF\_RX_{SS}(B)$, the receiver can correctly receive transmissions from A. However, when $RF\_RX_{SS}(B) > RF\_RX_{SS}(A)$, unlike the simultaneous transmission stage, the receiver does not receive any messages from B. This apparently contradictory behavior is not caused by the receiver's inability to correctly decode RF signals from B in the presence of weaker interfering RF transmissions from A, but is because of the RF packet format used. We explain this issue next.

Cricket software is written in the TinyOS software platform [100]. We used the TinyOS radio stack for RF packet transmission and reception in this experiment. Figure 4-26 shows the TinyOS RF packet format. The receiver detects the start of a RF

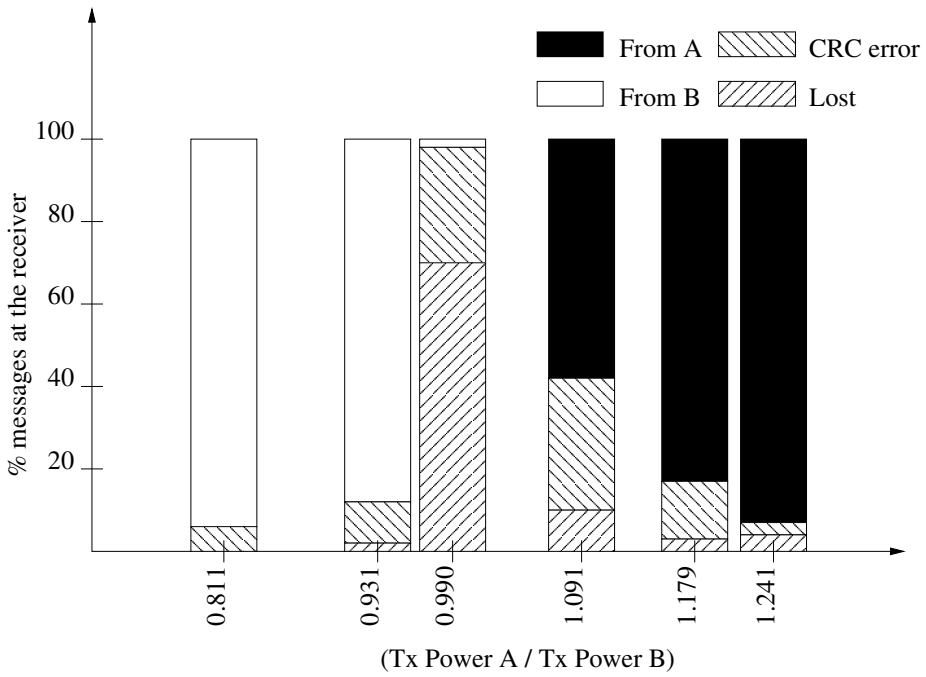


Figure 4-24: RF message delivery performance at a listener for simultaneous RF transmissions from two transmitters for different values of received RF signal strength ratio at the receiver.

message transmission after reading a number of bytes with a known preamble pattern (alternating 1s and 0s). Next, the receiver detects the start of message when it receives a known `START_OF_MESSAGE` byte. After this, the receiver reads `n` bytes, where `n = min(message_length, MAX_MESSAGE_LENGTH)`; n is limited to `MAX_MESSAGE_LENGTH` to overcome problems due to corrupted `message_length` field. Next, the receiver checks for bit errors by computing the packet's `CRC`. We can explain the results in Figure 4-25 as follows. When $RF\_RX_{SS}(B) > RF\_RX_{SS}(A)$, the receiver starts correctly decoding the weaker RF message from A; the receiver detects the preamble, the `START_OF_MESSAGE`, and then starts reading the number of bytes as indicated by `message_length`. However, at the middle of the packet, B starts transmitting. Since $RF\_RX_{SS}(B) > RF\_RX_{SS}(A)$, the receiver starts to decode the bytes from B's message (capture), until the receiver has read a total of n bytes. Since the n bytes read come from both A and B, the `CRC` fails, and the receiver flags this as a `CRC` error. Hence, due to the RF packet format, the receiver cannot correctly decode the stronger message in the presence of a partially overlapping weaker message, although the receiver can correctly decode the stronger message at the bit level.

We can solve the problem caused by a partially overlapping weaker RF message by modifying the receiver behavior as follows. The receiver, after receiving the preamble and the `START_OF_MESSAGE` field, continues to read the message as before. However, if the receiver receives a byte corresponding to the preamble (0x55 or 0xAA) it treats this byte as the preamble of a new RF message, and discards the current partially received message, and proceeds to process the *new* message. However, this approach

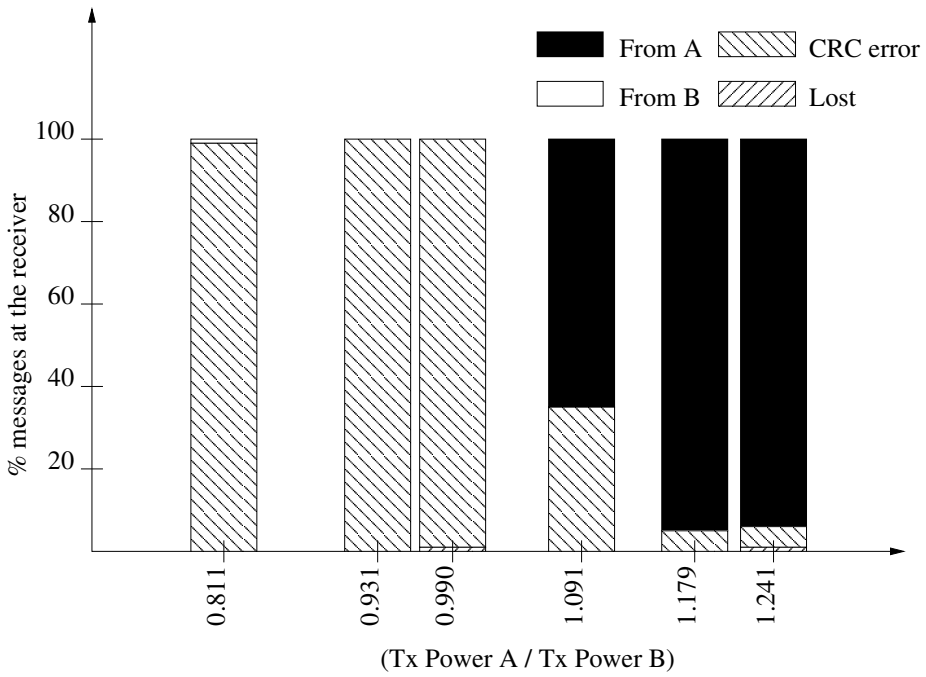


Figure 4-25: RF message delivery performance at a listener for RF transmissions from two transmitters that are delayed by half the packet duration for different values of received RF signal strength ratio.

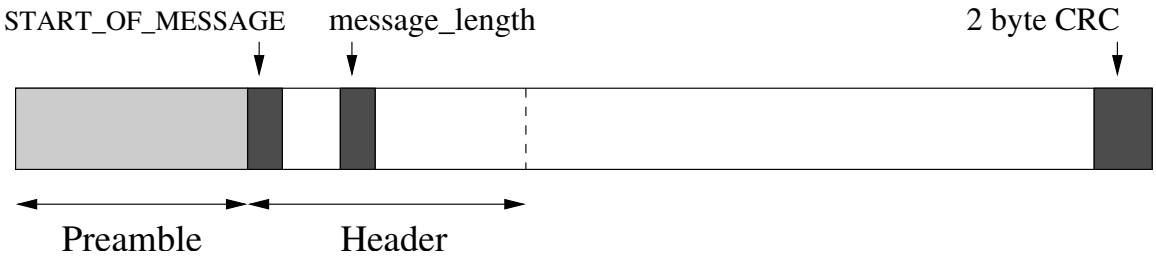


Figure 4-26: The TinyOS RF message format.

causes another problem. If the RF message contains data that corresponds to one of the valid preamble bytes, the receiver will incorrectly assume the start of a new message and discard the current message. We can use one of the following methods to overcome this problem.

- Select an RF message that does not contain preamble bytes. For example, we can use 7-bit ASCII messages with character 'U' (0x55) replaced with one of the control characters. This does not completely solve the problem since the two-byte `CRC` field may contain one of the preamble bytes. Having to select messages where the `CRC` field does not contain preamble bytes is inconvenient. However, we can get around this problem by treating valid preamble bytes in the `CRC` as always belonging to the `CRC`. Since the `CRC` field, located at the end of the packet, is only two bytes long, we lose at most three (due to bit offsets) bytes

from the start of the preamble of a new packet. We can compensate for this loss by making the preamble three bytes longer, since the receiver only needs a threshold number of valid preamble bytes to identify a new message.

- Use an escape character to escape the preamble bytes. In this approach we introduce an additional escape character, for example '\', before the each occurrence of a preamble byte or the escape character itself in the original message. The receiver removes these escape characters before processing any bits of the received RF message. The receiver assumes the start of a new packet if it receives a preamble byte without a preceding escape character. This approach is similar to the use of escape characters to escape control characters in text processing; for example we use the '\' character to escape control characters in LaTeXdocuments. It is also somewhat similar to the byte stuffing used in protocols like Serial Line IP (SLIP) and Point-to-Point Protocol (PPP) [77, 91]. In the escape character scheme, the RF message is larger than the original message because of the overhead from inserted bytes, and the length of the resulting message depends on the contents of the original message. This byte stuffing overhead can be significantly reduced by employing more complex byte stuffing algorithms [17].

- Encode the message using an alphabet that does not include preamble characters. In this scheme, the message, including the computed CRC, is encoded using some alphabet. For instance, we could use an encoding scheme such as Base16 [54], or a more compact encoding scheme such as a modified version of Base64, uuencode, or BinHex4.0 encoding. The encoding must replace the character corresponding to the preamble byte in the encoding alphabet with some other ASCII character [34, 33].

In our implementation, we modified the RF stack such that it treats preamble bytes in the middle of the received message as the start of a new message. We selected RF messages that do not contain any preamble bytes. We then repeated the experiment in Figure 4-23. Figures 4-27 and 4-28 show the message delivery performance at the receiver when the RF messages are transmitted simultaneously and when B's transmission is delayed by half a packet, respectively. With the modification to the RF stack, we observe that the receiver can correctly receive a delayed stronger RF message even when it partially overlaps with an interfering weaker RF message.

4.5 Deployment Considerations

4.5.1 Ultrasonic Noise

Since Cricket uses RF and US for distance measurements, ultrasonic noise in the environment will have an adverse effect on the system performance. We have observed the following common ultrasonic sources in an office environment: certain faulty

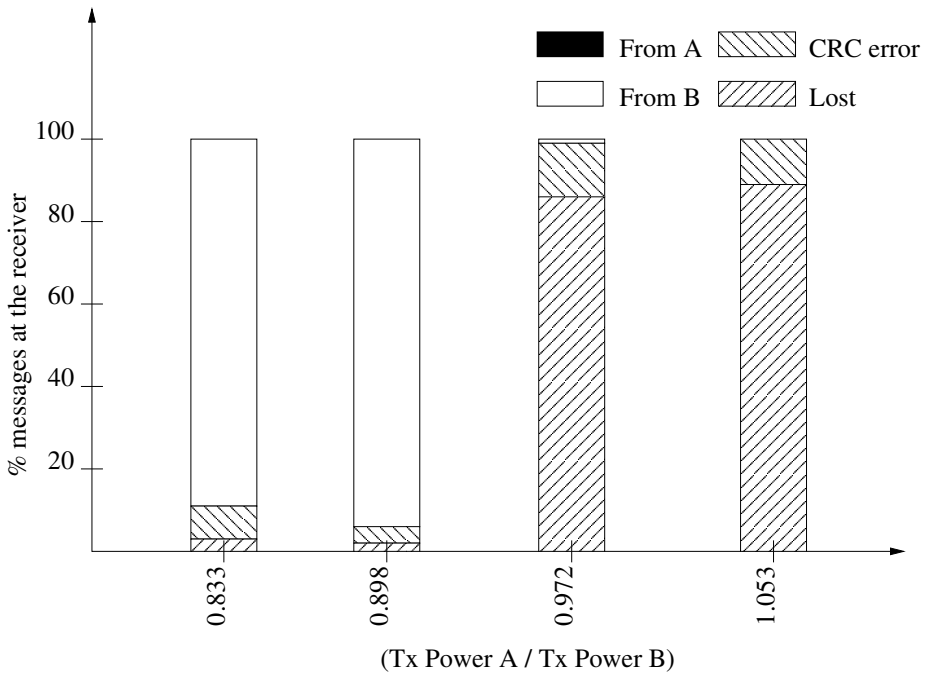


Figure 4-27: RF message delivery performance at a listener for simultaneous RF transmissions from two transmitters for different values of received RF signal strength ratio with the receiver starting a new packet upon the receipt of preamble bytes.

fluorescent lamps, jangling keys, air conditioners, and air conditioning ducts. Since the ultrasonic receiver is a physical resonator that resonates at 40 kHz, any loud noise such a banging door can cause the receiver to resonate at 40 kHz, which produces ultrasonic noise.

4.5.2 Line-of-Sight requirements

We have observed that it is comparatively easier to block the ultrasonic transmissions by placing the hand in front of the transmitter compared to blocking the signal at the receiver. Since ultrasonic signals can bend around objects, a person holding a listener does not completely block ultrasonic signals coming from a ceiling mounted beacon, provided the listener is held at a sufficient distance away from the body (the distance from the body determines how much the waves need to bend when reaching the receiver).

If the ultrasonic noise is infrequent, the outlier rejection algorithms can filter out the resulting incorrect distances. However, a continuous source of ultrasonic noise can severely degrade the Cricket performance.

4.6 Chapter Summary

This chapter described the algorithms for obtaining accurate beacon-to-listener distances in Cricket. This chapter examined the RF and US-based TDOA approach to

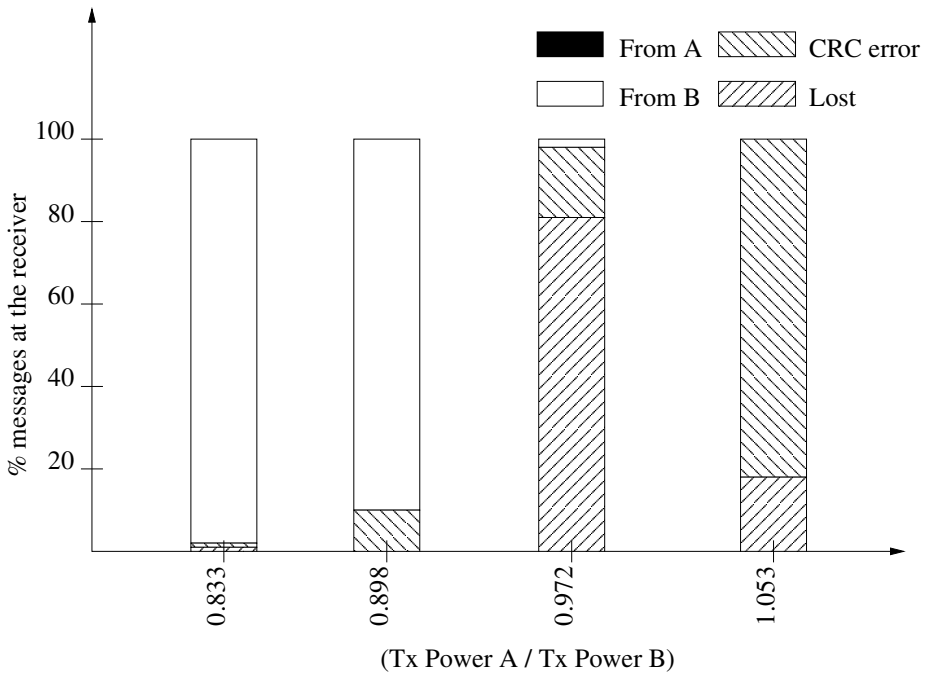


Figure 4-28: RF message delivery performance at a listener for RF transmissions from two transmitters that are delayed by half a packet duration for different values of the received RF signal strength ratio with the receiver starting a new packet upon the receipt of preamble bytes.

measure distances. It also described the beacon collision avoidance, listener collision detection, and outlier rejection algorithms that prevent incorrect distance samples at the listener due to uncoordinated beacon transmissions. This chapter used both simulations and experiments to determine the throughput of the Cricket system. This chapter also examined the RF scalability performance of Cricket. The next chapter describes how Cricket listeners use distances to nearby beacons to determine listener space, position, and orientation.

Chapter 5

Location Estimation Techniques

The previous chapter described how Cricket enables a listener to compute the distances to nearby beacons. Listeners use these distances to compute three different types of listener location information: the physical space, position, and orientation. This chapter describes the algorithms used to determine these three classes of location.

Section 5.1 describes how to deploy beacons to determine the current space that a listener is in, where "space" is defined as a human-readable label representing a region such as a room or a portion of a room. Some applications require finer-grained location information, and may benefit from position coordinates. Section 5.2 describes how to obtain position information in the form of the (x, y, z) coordinates of the listener, using the measured distances and the known (x, y, z) coordinates of the beacons. The third type of location information is orientation, which enables us to build "pose-aware" applications [99]. Section 5.3 examines how we can use listeners with multiple ultrasonic receivers to obtain orientation information by accurately measuring distance differences between multiple receivers. We first examine how to use phase difference measurements to accurately measure distance differences at multiple receivers. However, obtaining differential distance from phase differences results in *phase ambiguity* problem due to the periodicity of the received ultrasonic signals. Section 5.3 also describes two techniques for overcoming this problem, and show how to use phase differences to infer a listener's orientation.

5.1 Listener Space

"Space" is a natural form of location information for humans. For instance, the statement "I am in room 920 of the Stata Center at MIT" defines the position of the speaker to within some geographic space surrounded by a boundary. The boundary surrounding a particular space can be either real or virtual. A wall is a real boundary, while a space identified as "the north side of room" assumes a virtual boundary between the two sides. Cricket enables listeners to determine the space they are in, by using beacons to demarcate boundaries.

Cricket can accurately detect real boundaries such as walls because a real bound-

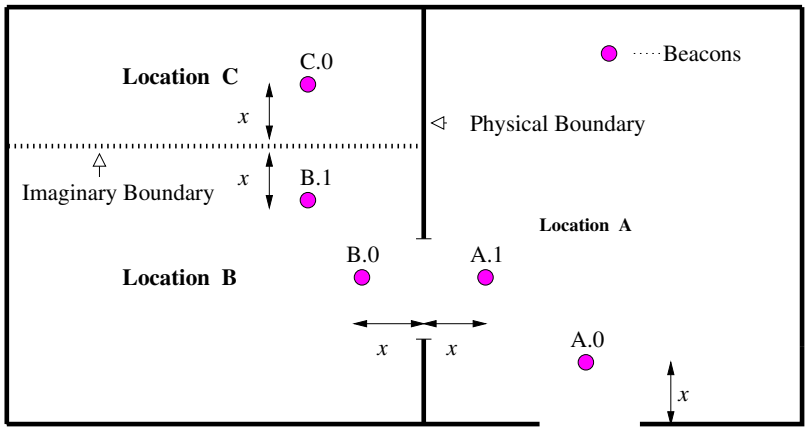


Figure 5-1: Correct positioning of beacons to detect boundaries.

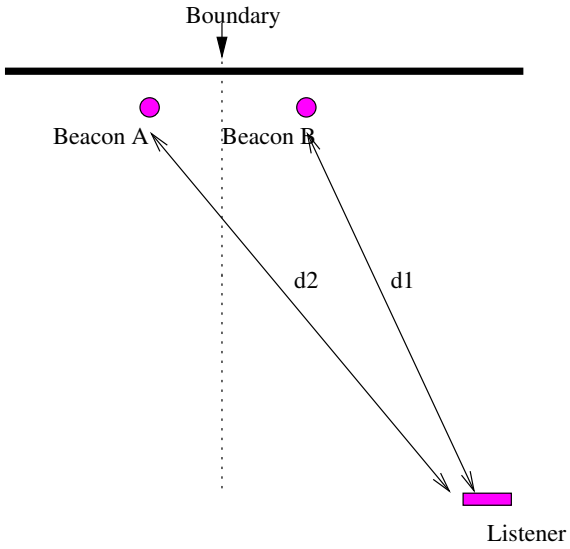


Figure 5-2: Closest beacon is always in the same space as the listener.

ary does not let ultrasound go through. To demarcate open boundaries that allow ultrasound to go through, we deploy a pair of beacons at equal distances away from each open boundary as shown in Figure 5-1. Once deployed, we program each beacon inside a particular space with the name of that space (or the "space id"). A Cricket listener, after obtaining distances to all the nearby beacons, associates itself with space id of the closest beacon. Since we deploy a pair of beacons at equal distances away from each open boundary, the closest beacon is guaranteed to be in the same space as the listener (fig:closestBeaconForSpace).

5.1.1 Cricket Boundary Detection Accuracy

We used the setup shown in Figure 5-3 to determine the Cricket boundary detection accuracy. We placed two beacons on the ceiling 120 cm apart. We selected a point 235 cm below the ceiling, and at equal distances from the ultrasonic transmitters on

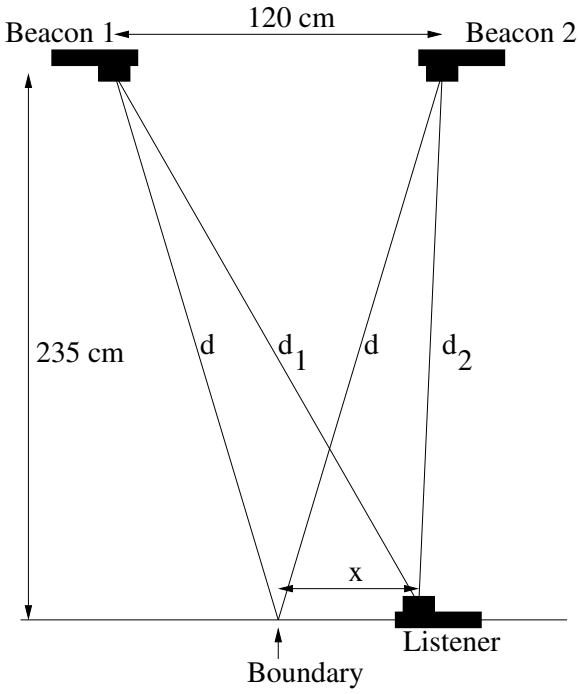


Figure 5-3: Experiment setup for measuring Cricket boundary detection accuracy.

the two beacons, as the boundary. We placed a listener at a distance x away from the boundary and collected distance measurements, d_1 and d_2, to the two beacons. We represented the distance by the number of processor clock ticks that the ultrasound signal takes to travel from a given beacon to the listener (1 clock tick $\simeq$ 1 μs). For a given x, we used 100 pairs of (d_1, d_2) values to determine the percentage of correct boundary decisions at the listener, where a correct boundary decision corresponds to $d_1 > d_2$, $d_1 = d_2$, and $d_1 < d_2$ for $x > 0$, $x = 0$, and $x < 0$, respectively. We varied x from -15 mm to $+15$ mm in increments of 5 mm.

Figure 5-4 shows the percentage of correct boundary decisions as a function of x. The listener can identify its placement with respect to the boundary 100% correctlyof the time for $|x| > 10$ mm. Hence, Cricket can detect open boundaries with an accuracy of about 1 cm. This experiment was conducted in the absence of any other beacons, without any reflections, and in the presence of a line-of-sight path between the listener and the closest beacon. In practice, these assumptions may not be valid all the time, so it is possible for the listener to make an incorrect decision about its current space.

We observe that Cricket listener can determine the closest beacon with a better accuracy than the basic ranging performance (Section 4.1). This improved accuracy is because determining the closest beacon requires only a distance comparison, compared to determining the actual beacon-listener distances. Distance comparison has better accuracy because, assuming that the beacons are located close to each other, the ultrasonic signals reaching the listener from both beacons are affected almost identically by environmental factors, and both ultrasonic signals are attenuated similarly because both beacons have almost identical distances and orientations with respect

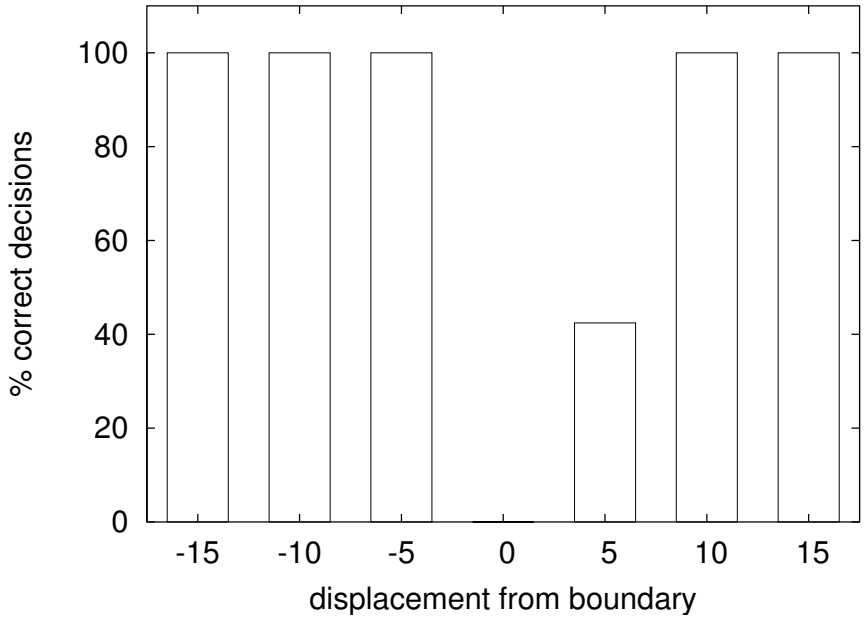


Figure 5-4: Cricket boundary detection performance results. The listener can correctly identify its placement w.r.t. boundary 100 % of the time, when the listener is placed at a distance $1cm$ or more from the boundary.

to the listener.

5.2 Listener Position

Some indoor applications require location information in the form of (x, y, z) coordinates of the device's current position. For example, when an application needs to know how two objects are positioned with respect to each other, it needs to know the location of each object as a set of coordinates defining one or more points on that object. A listener computes its position within the coordinate system defined by Cricket, using the distances to multiple beacons and known beacon coordinates. In this discussion, we assume that beacon coordinates are known. Chapters 6 and 7 describe techniques by which beacons obtain their coordinates.

We first show how to estimate the position of a static listener. Consider a listener located at (x_l, y_l, z_l) in the beacon coordinate system. Assume that the listener can measure the distances to n beacons $b_1, b_2, \ldots, b_n$. Let d_i be the *measured* distance between b_i and the listener. Beacon b_i has coordinates (x_i, y_i, z_i). The *true* distance between the listener and b_i is given by $d_i - \epsilon_i$, where ϵ_i is the measurement error. If $n = 3$, and the distance measurement errors are not too large, we can obtain a reasonable estimate of the listener position by solving the three simultaneous equations ${d_i}^2 = (x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2$ for $i = 1, 2, 3$.

We get two possible solutions for these three equations, one with the listener located above the plane containing the three beacons and the other with the listener

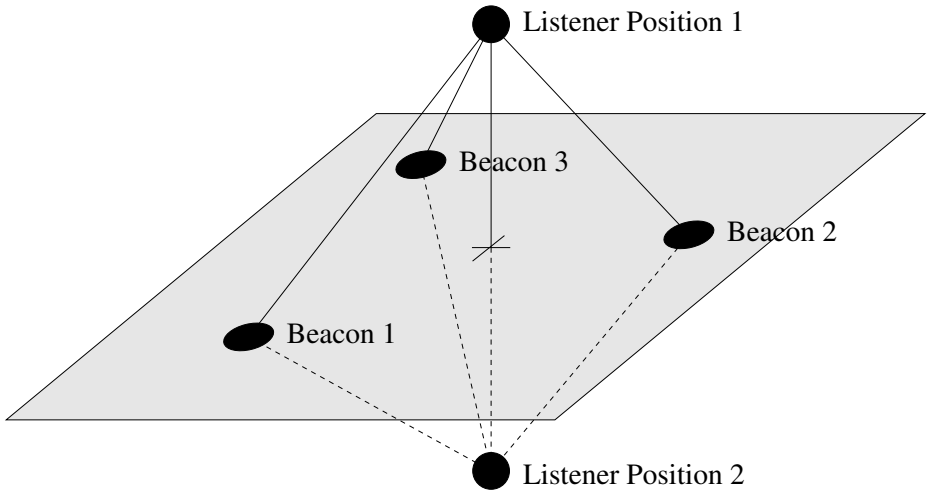


Figure 5-5: There are two possible listener positions that satisfy the distances to three beacons. These listener positions are at equal distances from the plane containing the three beacons.

below this plane, as shown in Figure 5-5. We can uniquely determine the listener position provided we know where the listener is located with respect to the plane containing the three beacons; for instance, if the beacons are deployed on the ceiling, we can assume that the listener is always located below the plane containing the beacons.

When $n \geq 4$, we can use the following non-linear optimization to compute listener coordinates. We assign some initial coordinates (x_0, y_0, z_0) to the listener. For each beacon i, we define a residual $e(i)$ as follows.

$$e(i) = \sqrt{(x_0 - x_i)^2 + (y_0 - y_i)^2 + (z_0 - z_i)^2} - d_i$$

We define sum squared error E_{ss} as

$$E_{ss} = \sum_{i=1}^{n} e(i)^2.$$

The optimization problem is to find the listener coordinates (x_0, y_0, z_0) that minimizes E_{ss}. We can use the following scheme to efficiently find the global minimum. First, we use some three distances to compute two possible solutions to the listener position. Next, we compute E_{ss} for both these positions; if the n beacons are not co-planar, under the assumption of small measurement error ϵ_i, one of these two listener positions will be closer to the true listener position compared to the other. The position closer to the true listener position will have a smaller E_{ss}. We then use a standard nonlinear optimization algorithm with the listener position corresponding to the smaller E_{ss} value as the initial coordinate assignment. If all the beacons are coplanar, then we obtain two possible solutions for the listener position, irrespective of the number of beacons, when $n > 3$. Similar to the $n = 3$ case, we have to use additional information to uniquely identify the position of the listener.

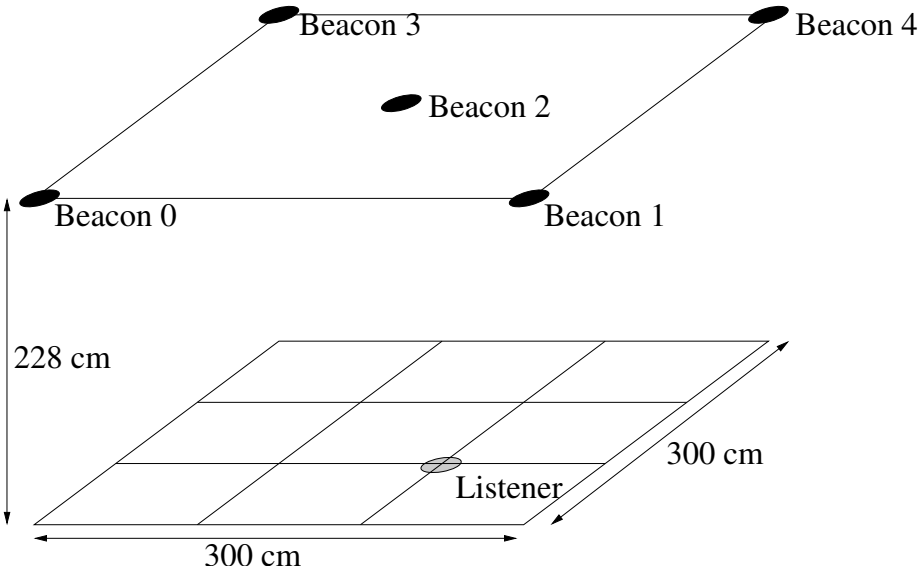


Figure 5-6: Experiment setup for measuring Cricket position estimation accuracy. Distance samples were collected by placing the listener on a square grid parallel to the plane containing the beacons.

We used the setup shown in Figure 5-6 to measure the position estimation accuracy of the Cricket system. We placed five beacons as shown, and collected distance samples by placing the listener at 16 points on a plane parallel to the plane containing the beacons. We computed the listener position using the distance samples to the closest three beacons. Figure 5-7 shows the position offset (the Euclidean distance between the true position and the estimated position) for different listener positions. We observe that for the majority of cases the position estimation error is less than 10 cm.

When the listener is mobile, the multiple beacon distances are received at different instances of time, when the listener is at different positions. However, we can still compute a representative position for the listener using these distance samples. We use the same technique as for the static listener, the only difference being that the error ϵ_i now has two components; the measurement error, and error caused by the listener being at different positions when the different distance samples are obtained.

5.2.1 Concurrent Position and Speed of Sound Estimation

In the previous section, we examined how to determine listener position using the distances to multiple beacons. The listener obtains these distances by multiplying the time taken for the ultrasonic signal to reach the listener by the speed of sound. However, as discussed in Section 4.1.1, the speed of sound depends on environmental factors such as temperature, humidity, and atmospheric pressure. Although beacons and listeners use temperature sensors to compensate for the variations of speed of sound due to temperature, other environmental factors and the finite resolution in temperature measurements can result in the actual speed being different from that

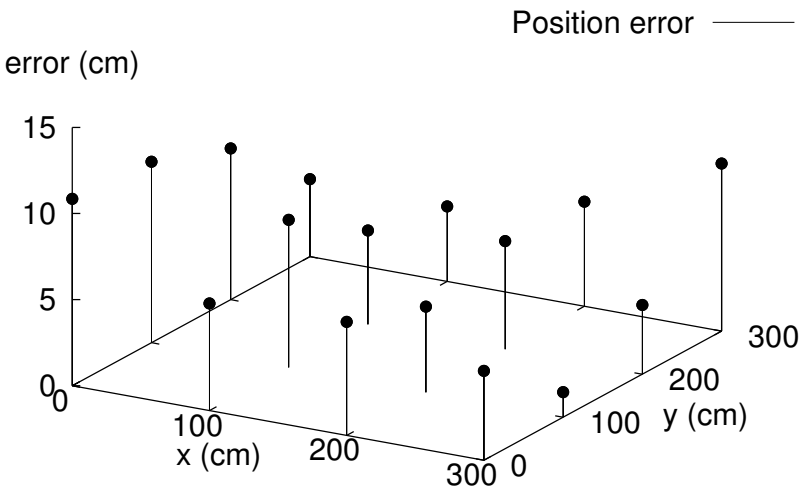


Figure 5-7: The position estimation error at different listener positions, when position calculated from the measured distances to the three closest beacons.

used by listeners to compute distances. This section describes a technique for computing listener position and the speed of sound simultaneously, using the ultrasound propagation time from four coplanar beacons.

Consider four coplanar beacons $b_1 \ldots b_4$. Let t_i denote the time taken for the ultrasonic signal from the beacon b_i to reach the listener. The listener estimates t_i by measuring the time difference of arrival of the RF and the ultrasound signals from beacon b_i. We denote the estimated value of t_i by $\hat{t}_i$. Let v denote the speed of sound. Assuming that the plane containing the beacons represent $z = 0$ plane, we denote the coordinates of each beacon b_i and the listener by $(x_i, y_i, 0)$ and (x_l, y_l, z_l), respectively. We can write the following family of equations to determine the four unknowns (x_l, y_l, z_l, v).

$$(x_l - x_i)^2 + (y_l - y_i)^2 + z_l^2 = v^2 t_i^2, \;\; 1 \leq i \leq 4 \tag{5.1}$$

We eliminate z_l^2 from these equations by subtracting each equation from the previous one, and obtain the following three *linear* equations with the three variables, x_l, y_l and v^2:

$$A \times \begin{bmatrix} x_l \\ y_l \\ v^2 \end{bmatrix} = \begin{bmatrix} x_1^2 - x_0^2 + y_1^2 - y_0^2 \\ x_2^2 - x_1^2 + y_2^2 - y_1^2 \\ x_3^2 - x_2^2 + y_3^2 - y_2^2 \end{bmatrix} \tag{5.2}$$

where the matrix A is given by

$$A = \begin{bmatrix} 2(x_1 - x_0) & 2(y_1 - y_0) & (t_1^2 - t_0^2) \\ 2(x_2 - x_1) & 2(y_2 - y_1) & (t_2^2 - t_1^2) \\ 2(x_3 - x_2) & 2(y_3 - y_2) & (t_3^2 - t_2^2) \end{bmatrix}.$$

Since only the estimated value, $\hat{t}_i$, of each t_i is available, we use the estimated matrix, $\hat{A}$, obtained by replacing each t_i in A by $\hat{t}_i$, to compute the values of x_l, y_l and v^2. If the determinant of $\hat{A}$ is non-zero, we can solve the Equation 5.2 to determine unique values for x_l, y_l, and v^2. We can substitute these values back in to the Equation 5.1 to obtain the value of z_l^2. The positive square root of z_l^2 gives z_l, assuming (without loss of generality) that $z > 0$ for points below the beacons.

If there are m ($m > 4$) beacons within the listener's range, we obtain a overconstarined system of equations which may improve the listener position estimation performance. If $\epsilon i = v^2 t^2 - v^2 \hat{t}_i^2$, then we can rewrite the Equation 5.1 as follows,

$$(x_l - x_i)^2 + (y_l - y_i)^2 + z_l^2 = v^2 \hat{t}_i^2 + \epsilon i, \;\; 1 \leq i \leq m, m > 4. \tag{5.3}$$

We again eliminate z_l^2 from these equations by subtracting each equation from the previous one and discarding the last equation. This results in $(m-1) > 3$ *linear* equations with the three variables, x_l, y_l and v^2. We solve the resulting overconstrained system of linear equations using a least-square solver that minimizes the squared error δ [61], where,

$$\delta = \sum (\epsilon i - \epsilon_{i+1})^2. \tag{5.4}$$

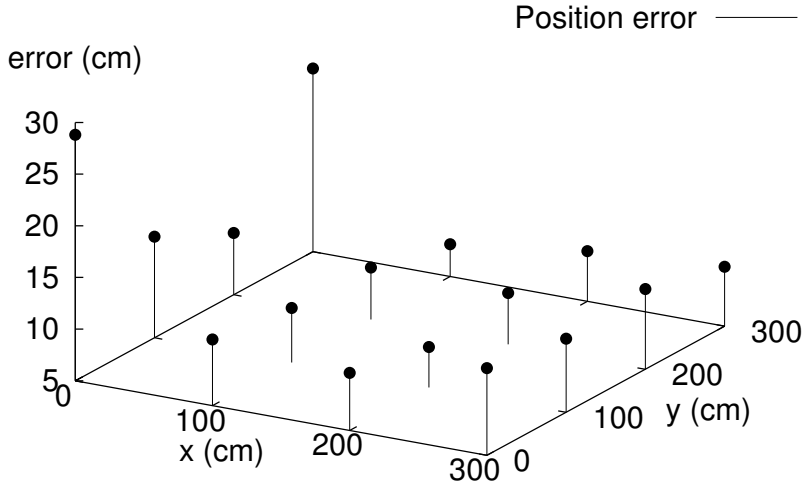


Figure 5-8: The position estimation error at different listener positions, when both the listener position and the velocity of sound are calculated from the measured distances.

We used the data collected in the experiment setup in Figure 5-6 to calculate both the listener position and the velocity of sound by substituting the distances to the four closest beacons to the Equation 5.2. Figure 5-8 shows the position estimation error at different listener positions.

We observe that computing both the listener position and the speed of sound using measured distances results in a larger position estimation error compared to position estimation from temperature-compensated distance measurements. We can explain this increased error as follows. Both position estimation techniques described above measures four variables. The temperature-compensated distances approach measures three distances and the temperature, while the pure timing-based approach measure four values that are proportional to four distances. The percentage error in temperature measurement is smaller than the percentage error in timing the distances, because the relatively low temperature coefficient enables us compute the the speed of sound accurately. Hence, the temperature-compensated approach with one accurate measurement performs better.

5.2.2 Kalman Filter-Based Listener Position Estimation

We have also designed and implemented a Kalman filter-based approach to overcome the problems associated with non-simultaneous distance samples at a mobile listener [92]. Using a Kalman filter, the listener maintains an estimate of its state (e.g., its position and velocity). When the listener receives a distance sample from a beacon, the listener uses the error between its current position estimate and the set of the possible locations as determined by the new distance sample, to update its current position and velocity estimates. This computation is the correction phase of the Kalman filter. If the error between estimated distance and the measured distance is too large, then the distance sample is assumed to be incorrect. However, distance measurements with small distance errors that are not identified as incorrect samples may cause the Kalman filter to reach a bad state. When the filter is in a bad state, it will treat almost all the distance samples as incorrect, thus preventing it from getting out of the bad state. Because the fraction of incorrect distance samples in Cricket is small, the filter can identify that it is in a bad state when it identifies a sequence of distance samples as incorrect. Once the filter identifies a bad state, it needs a mechanism to reset itself by accurately computing its current position. To accurately compute the current position, the listener enters an *active listener mode* where it actively transmits and request multiple simultaneous distance samples from nearby beacons.

In the active listener mode, a listener actively transmits an RF and ultrasonic message similar to the beacon transmissions used for computing distance estimates at a listener. As explained below, these transmissions do not reveal the identity of the listener. The beacons, upon receiving the RF and ultrasonic messages from a listener, compute the distance to the listener and send this information back to the listener using an RF message. As long as the active listener mode is infrequent, it does not significantly degrade scalability [92].

In their normal mode of operation, to conserve energy, the beacons turn off the

```
transmit_message(){
   sendRFmessage();
   turn_on_RF_rx();
   start_sleep_timer();
   ...
}
sleep_timer_expired(){
   if not running_stay_up_timer then
      turn_off_RF_rx();
   endif
}
stay_up_timer_expired(){
   turn_off_RF_rx();
}
RF_message_received(message){
   if message.type == STAY_UP_REQUEST then
      start_stay_up_timer();
   endif
   ...
}
```

Figure 5-9: Pseudocode of a beacon responding to a STAY_UP_REQUEST from an active listener.

RF and ultrasonic receivers after transmitting each beacon message. Because of this, an active listener requesting distance estimates from beacons needs a mechanism to request the nearby beacons to keep their RF receivers and ultrasonic receiver circuits turned on until the listener transmits its message.

The listener, before transmitting its message, transmits a STAY_UP_REQUEST message to its nearby beacons, one beacon at a time, asking them to keep the RF and ultrasonic receivers turned on for some time. To enable the listener to send the STAY_UP_REQUEST, the beacons keep their RF receivers on for a short duration after transmitting a beacon message. The listener sends its STAY_UP_REQUEST message shortly after receiving a beacon message from a beacon. The listener uses a random delay before transmitting the STAY_UP_REQUEST message to prevent collisions among requests from multiple listeners.

Figure 5-9 shows the pseudocode that handles the STAY_UP_REQUEST messages at the beacon. To receive STAY_UP_REQUEST messages from listeners, the transmit_message() routine on the beacon turns on the RF receiver and starts the sleep_timer after transmitting the beacon RF message. If the beacon receives a STAY_UP_REQUEST before the sleep_timer expires, the beacon starts a longer stay_up_timer. Any STAY_UP_REQUEST messages received before the stay_up_timer expires resets the timer. After sending STAY_UP_REQUEST messages to nearby beacons, the listener transmits an RF message and the corresponding ultrasonic signal. The listener sends a random nonce in the ID field of the ranging message rather

```
process_active_listenr_rqst(nonce){
   compute_distance();
   for i=1 to 10 do
      d1 = 1.25*PACKET_DURATUION*uniform_int(0,9);
      d2 = 0.25*PACKET_DURATUION*uniform(0,1.0);
      delay(d1 + d2);
      if carrier_free() then
         send_reply(nonce);
         break;
      endif
   endfor
}
```

Figure 5-10: Pseudo code of a beacon responding to an active listener distance request

than revealing its own ID. The beacon, after receiving this message from a listener, computes the distance to the listener, and transmits an RF message containing the computed distance and the nonce. The RF messages from beacons and active listeners use different message types to prevent beacons inadvertently replying to RF messages from other beacons. Figure 5-10 shows the pseudocode that processes the ranging message at the beacon. Because multiple beacons typically reply to an active listener message, they may collide at the listener.

Each beacon uses a random delay before its reply to reduce these collisions. Each beacon selects a random time-slot, out of 10 time slots of duration $1.25 \times T_R$ each, where T_R is the duration of the reply message. To prevent collisions among beacons that select the same time slot, each beacon computes a random `offset` which is uniformly distributed between 0 and $0.25T_R$. A beacon that selects the nth timeslot transmits it reply after a delay of $(1.25 \times n \times T_R + \texttt{offset})$.

5.3 Listener Orientation

Cricket also provides orientation information. As discussed in Section 2.5, traditional approaches to determining orientation have either poor indoor performance or tend to be expensive. Traditional approaches usually rely on the availability of an external frame of reference such as the earth's magnetic poles to determine the orientation. In contrast, when we deploy a local location system using Cricket, we may prefer our frame of reference for orientation to be consistent with the local coordinate system. This section describes how to enhance the Cricket listener such that it can obtain orientation within the local coordinate system defined by the Cricket beacons [72].

We first examine how to obtain the orientation of an object from the orientation of line segments on that object. Consider the line segment joining the two points (P_0, P_1) on the object in Figure 5-11. If we know the orientation of the vector along (P_0, P_1) in 3D space, the object should be oriented such that the line joining these two points are parallel to this vector. However, as the Figure 5-11 shows, due to

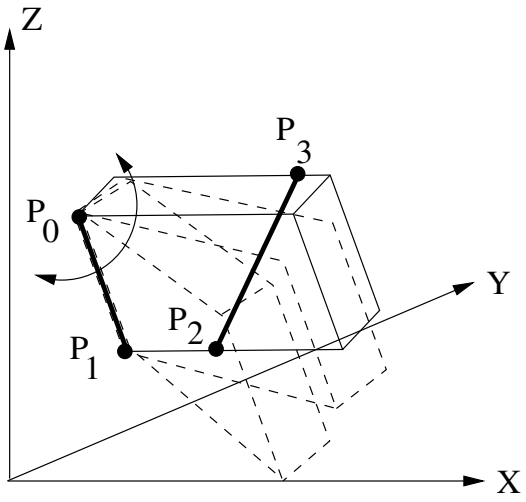


Figure 5-11: The orientation of two line segments that are not parallel to each other (and non-coplanar for 3D objects) uniquely determine the orientation of an object in 3D space.

symmetry, the object can still rotate around this line; hence a single line segment does not uniquely identify the orientation of an object in 3D space. If we know the orientation vector of the line joining two other points, (P_2, P_3), such that the line (P_0, P_1) is not parallel to the line (P_2, P_3), then we can determine the orientation of the object uniquely, since the line segment (P_2, P_3) restricts the rotation of the object around (P_0, P_1). Hence, we can uniquely determine the orientation of a 2D or a 3D object from the orientation of two line segments that are not parallel to each other. We next focus on obtaining the orientation of a line segment in 3D space.

5.3.1 Obtaining Orientation form Distances

Our approach is to place multiple ultrasonic receivers on a Cricket listener and determine the orientation of the line segments joining these receivers. Consider a listener with two ultrasonic receivers attached to it. One way to compute the orientation of the line segment joining the two receivers is to compute the positions of the two receivers using the techniques discussed above; using these positions we can compute the orientation of the vector along these two points, as shown in Figure 5-12. However, the magnitude of the position estimation error in Cricket, compared to the physical size of a typical hand held device, makes this approach impractical.

As an example, consider two ultrasonic receivers R_1 and R_2, that are at a distance L apart, shown in Figure 5-13; here, position estimation errors cause the estimated position of R_1 to be offset by a distance d_e with respect to its true position. If $d_e \ll D$, the maximum error in estimated orientation, θ_e (in radians) is given by:

$$\theta_e \simeq \frac{d_e}{L}.$$

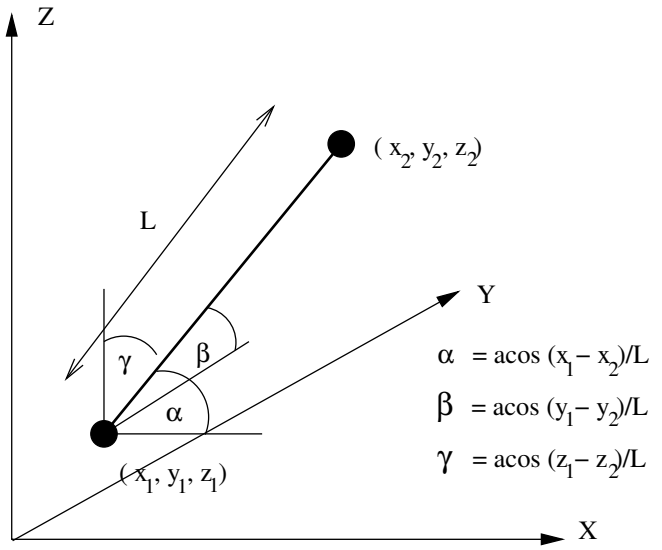


Figure 5-12: The orientation of a line segment can be obtained from the coordinates of the endpoints.

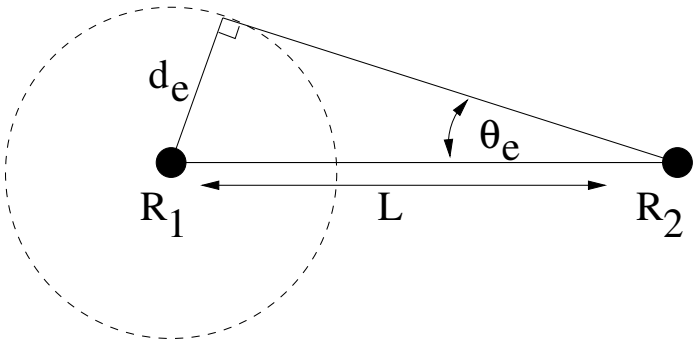


Figure 5-13: When the positions of the two receivers R_1 and R_2 used to determine the orientation of the line segment (R_1, R_2), a position estimation error of d_e at R_1 results in a maximum orientation estimation error of θ_e.

In Cricket, $d_e \simeq 10$ cm; hence, an orientation accuracy of $\theta_e < 1^0$ requires $L > 570$ cm, which is much larger than the dimensions of a typical handheld device.

Estimating the *differential* distance between R_1 and R_2 instead of the position estimates of R_1 and R_2 significantly improves the accuracy of the orientation estimate. Consider an ultrasonic signal from a far-away beacon being received at two receivers R_1 and R_2 (Figure 5-14). The perpendicular to the line joining R_1 and R_2 is at an angle θ to the direction of the ultrasonic transmitter (we assume that the ultrasonic transmitter, R_1, and R_2 lie on the plane of Figure 5-14). Let the distances from the transmitter to R_1 and R_2 be d_1 and d_2 respectively. Let $\delta d = d_1 - d_2$ and assume $d_1 \gg L$ and $d_2 \gg L$. Then,

$$\theta \simeq \arcsin \frac{\delta d}{L}. \tag{5.5}$$

We can measure the differential distance δd with a higher accuracy than the indi-

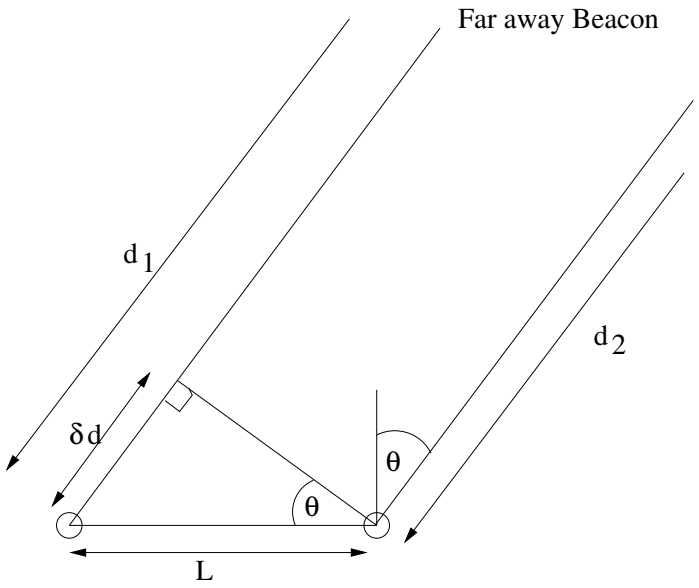


Figure 5-14: The angle θ between a receiver pair and a beacon can be obtained from the distance difference $d_1 - d_2$ and the separation L of the receiver pair.

vidual distances d_1 and d_2, for two reasons. First, individual distance measurements depend on environmental factors such as the humidity and the ambient temperature (Section 4.1.1). Although these factors are not uniformly distributed across a room, for small values of L, these factors have an almost identical impact on signals received at R_1 and R_2 from a given source. Second, in Section 4.1.2, we observed that distance estimate errors depend on the received signal strength; the received signal strength at R_1 and R_2 should be almost identical, because they both face the same direction and are nearly co-located.

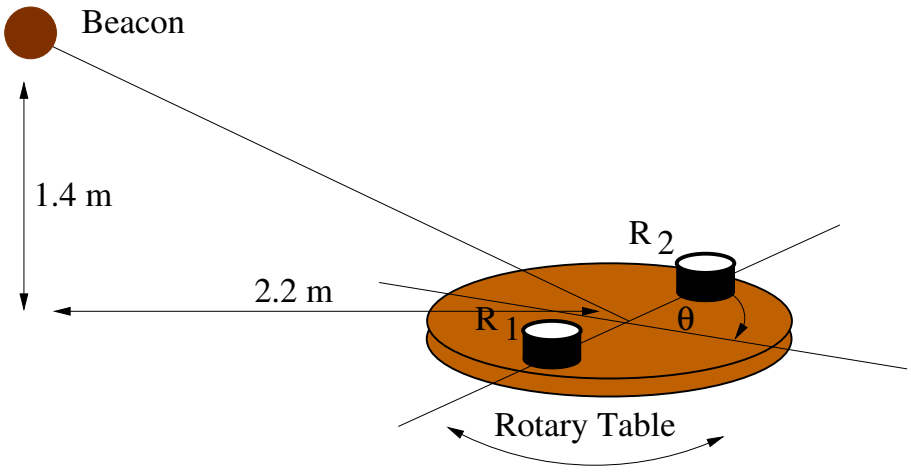


Figure 5-15: The setup for measuring distance difference to beacon B from two receivers R_1 and R_2.

We used the experimental setup shown in Figure 5-15 to determine the differential

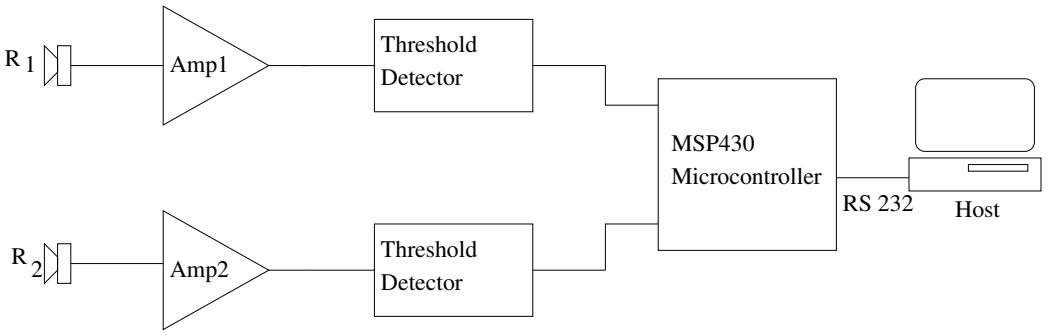


Figure 5-16: The block diagram for measuring the time interval δt between US signal arrival at R_1 and R_2 in Figure 5-15.

distance measurement accuracy in Cricket. We attached two ultrasonic receivers, R_1 and R_2, to a rotating platform (a rotary table), to vary the angle θ between the line (R_1, R_2) and the beacon B. Figure 5-16 shows the block digram for measuring the time interval δt between the ultrasonic signal arrivals at R_1 and R_2. The two ultrasonic signals received at R_1 and R_2 were first amplified and then were converted to digital signals, which activated two "capture" pins on the MSP430 microcontroller [65]. These capture pins captured the value of an internal timer of MSP430. The time difference of arrivals, δt, of the ultrasonic signals was obtained from the difference of these captured values. The measured δt values were sent to the attached host.

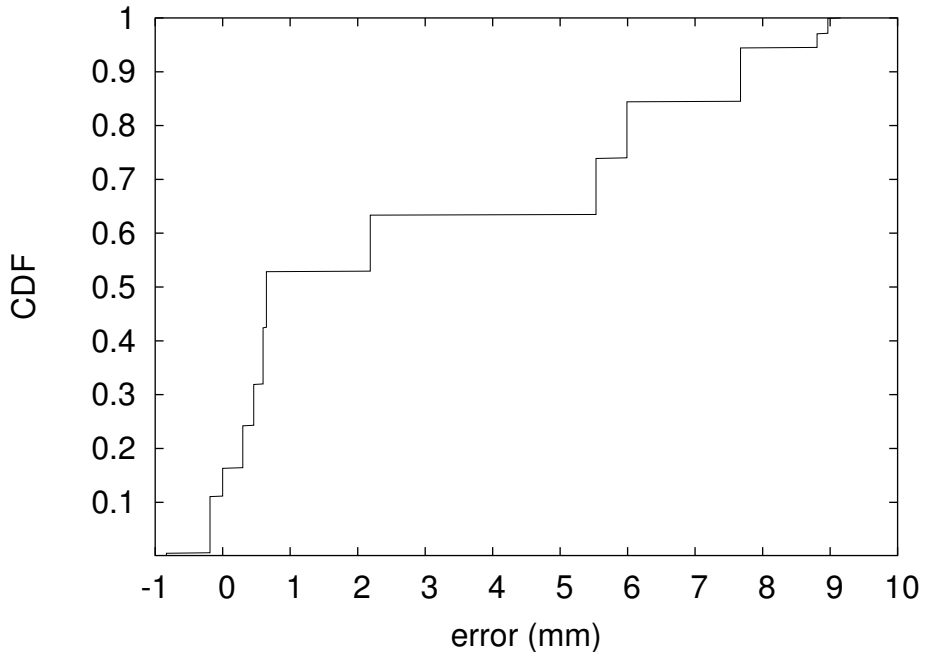


Figure 5-17: The estimated CDF of the measured differential distance δd, between R_1 and R_2 of Figure 5-15.

We measured δt for θ in the range $(-90^o, 90^o)$ in increments of 10^o. For each value

of θ, we collected 50 δt samples. We obtained the measured differential distance, δd, from R_1 and R_2 to the beacon B by multiplying δt by the speed of sound. We also computed the differential distances using the distances and the angles in the experimental setup. Figure 5-17 shows the CDF of the δd measurement error, defined as the difference between the measured value and the computed value. We observe that, although the differential distance estimation error is less than both position estimation error and the distance estimation error (Figure 5-7 and Figure 4-2), the differential distance measurement error can be as high as 9 mm.

Although differential distance measurement has higher accuracy than computing the node positions separately, it still has significant measurement error, since the ultrasonic receivers and detection circuits are not identical due to manufacturing tolerances. For instance, when δd has a measurement error of 9 mm, for $\theta = 0^0$, to achieve an angle estimation error $< 1^o$, $L > 52$ cm (from Figure 5-13), which is still too large for a typical handheld device.

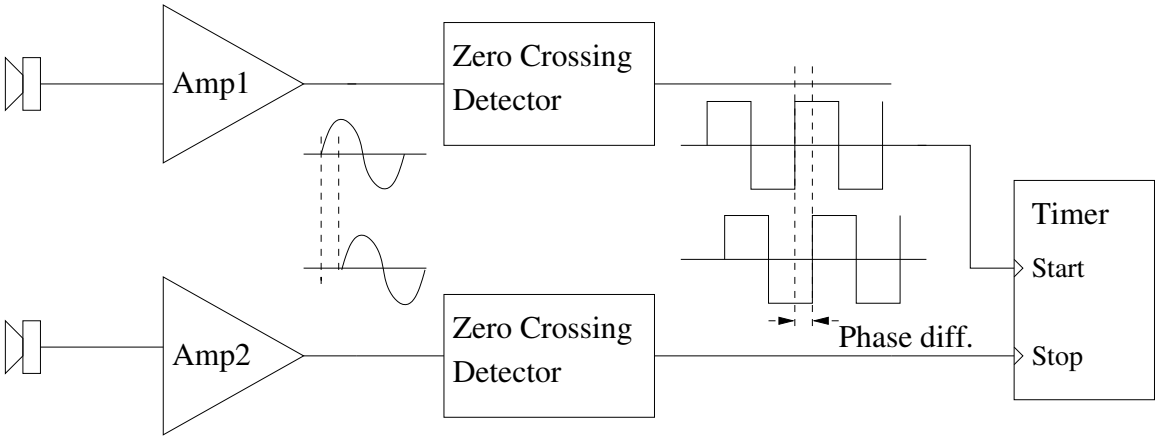


Figure 5-18: The setup used to measure the phase difference between the signals received at two ultrasonic receivers.

We can measure differential distance more accurately using the *phase difference* of the received signals. Figure 5-18 shows the setup used to measure the phase difference of the received signals. The 40 kHz sinusoidal signals at the two ultrasonic receivers R_1 and R_2 are amplified by the amplifiers `Amp1` and `Amp2`. Next, the amplified signals are converted to two 40 kHz square waves using *zero-crossing detector* circuits that detect signals going from -ve to +ve and vice versa. The phase shift between the positive edges of these square waves represents the phase difference between the original sinusoidal signals.

This scheme enables us to measure the phase difference between the received signals with a higher accuracy than the difference in arrival times, for the following reasons. When measuring phase difference, we only need to detect the zero crossing points of the signals. This can be done when the signals have fully developed at the receiver; in contrast, for arrival time measurements, we have to interpret the signal as it "ramps up" at the receiver. Robustly detecting the starting point of the signal is impeded by variations in the received signal strength, imperfections in

the detection circuits, and by receiver sensitivity characteristics. Unlike arrival time measurements, with phase difference measurements,we obtain multiple measurements per received signal because we observe multiple cycles of the signal.

We used the experimental set up shown in Figure 5-15 to determine the phase difference-based differential distance measurement accuracy. We placed two receivers R_1 and R_2 at 1.5λ apart, and we used the block digram shown in Figure 5-18 to measure the phase difference between the two receivers. We used the knowledge of the calculated differential distance between R_1 and R_2 to resolve the *phase ambiguity* in phase difference-based differential distance estimation (Section 5.3.2).

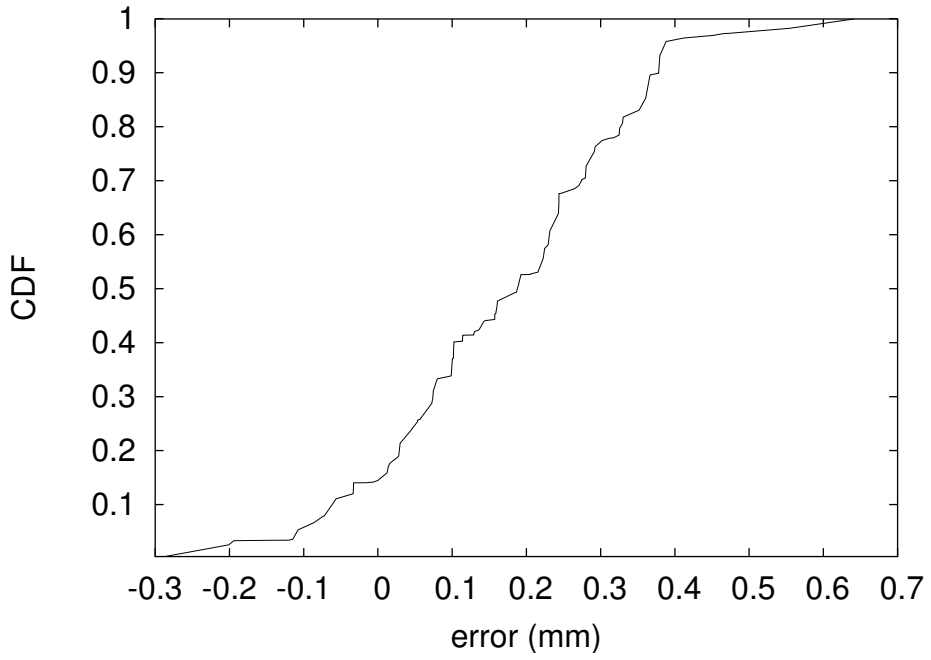


Figure 5-19: The estimated CDF of the phase difference-based differential distance measurement error.

We measured δt for θ in the range $(-90^o, 90^o)$ in increments of 5^o. For each value of θ, we collected 50 δt samples. We converted the measured δt values to corresponding δd values by multiplying them by the speed of sound. Figure 5-19 shows the CDF of the δd measurement error, defined as the difference between the measured value and the computed value. We observe that the phase difference-based differential distance estimation error is less than 0.65 mm.

5.3.2 The Phase Ambiguity Problem

Although measuring phase difference enables us to measure differential distance accurately, it suffers from *phase ambiguity* when we try to measure differential distances outside the range $(-\frac{\lambda}{2}, \frac{\lambda}{2})$. For example, because of the periodic nature of the signals, in Figure 5-18 we observe that, for $\delta d < \frac{\lambda}{2}$, a distance difference of δd and $\delta d + \lambda$ both result in a *visible* phase difference of δd. Hence, to prevent phase ambiguity we

need to make sure that the distance difference $|\delta d| < \frac{\lambda}{2}$. To ensure that for all values of *theta* in the range $(-\pi, \pi)$, the separation between the two receivers L must be less than $\frac{\lambda}{2}$. For a 40 kHz ultrasonic waveform at a temperature of 25°C and 50% humidity, $\lambda/2 = 4.35$ mm. This value is smaller than the diameter of the ultrasonic receivers used in the Cricket listeners (which are about 1 cm). Hence, we cannot physically place two receivers such that $L < \frac{\lambda}{2}$. The next two sections present two different techniques for overcoming the phase ambiguity associated with phase difference-based differential distance estimation, in which the ultrasonic receivers need not be placed $< \frac{\lambda}{2}$ apart.

5.3.3 Phase Disambiguation Using a Pair of Phase Differences

One way to tackle the phase ambiguity is to carefully place *three* receivers along a line, as shown in Figure 5-20, and use the *pair* of observed phase differences to estimate the actual distance difference. We can show that for certain values of inter-receiver distances, L_{12} and L_{23}, the actual phase difference between receivers R_1 and R_2 (say) can be disambiguated by the phase difference between receivers R_2 and R_3.

Let ϕ_{12} and ϕ_{23} be the actual phase differences of a beacon's waveform between receivers 1 and 2 and receivers 2 and 3, respectively. Then,

$$\phi_{ij} = 2n_{ij}\pi + \alpha_{ij}$$

for each pair of receivers (i, j), where n_{ij} are integers and $-\pi < \alpha_{ij} \leq \pi$. Because the actual phase difference between two receivers is proportional to the distance traversed by the signal from the beacon to each of the receivers, $\phi_{23}/\phi_{12} = (d_2 - d_3)/(d_1 - d_2) \approx L_{23}/L_{12}$ when $d_i \gg L_{ij}$.

What we will show is that it is possible to pick L_{12} and L_{23} such that one can use two sets of observed phase differences α_{12}, α_{23} to *unambiguously* estimate the actual phase difference ϕ_{12}. In particular, we show the following result: *If L_{12} and L_{23} are relatively prime multiples of $\lambda/2$, then it is possible to use α_{12} and α_{23} to unambiguously obtain the actual phase differences ϕ_{12} and ϕ_{23}.*

We argue this result by contradiction. Suppose in fact there are two possible actual phase differences corresponding to a given observed phase difference for each receiver. For pair (i, j), call these differences ϕ'_{ij} and ϕ''_{ij}. Then, the following sets of equations hold:

$$\phi'_{ij} = 2n'_{ij}\pi + \alpha_{ij},$$
$$\phi''_{ij} = 2n''_{ij}\pi + \alpha_{ij}.$$

Because each observed ϕ_{12} is related to the corresponding ϕ_{23} by the ratio L_{23}/L_{12}, the above equations can be rewritten as:

$$2n'_{23}\pi + \alpha_{23} = (L_{23}/L_{12})(2n'_{12} + \alpha_{12}), \tag{5.6}$$

$$2n''_{23}\pi + \alpha_{23} = (L_{23}/L_{12})(2n''_{12} + \alpha_{12}). \tag{5.7}$$

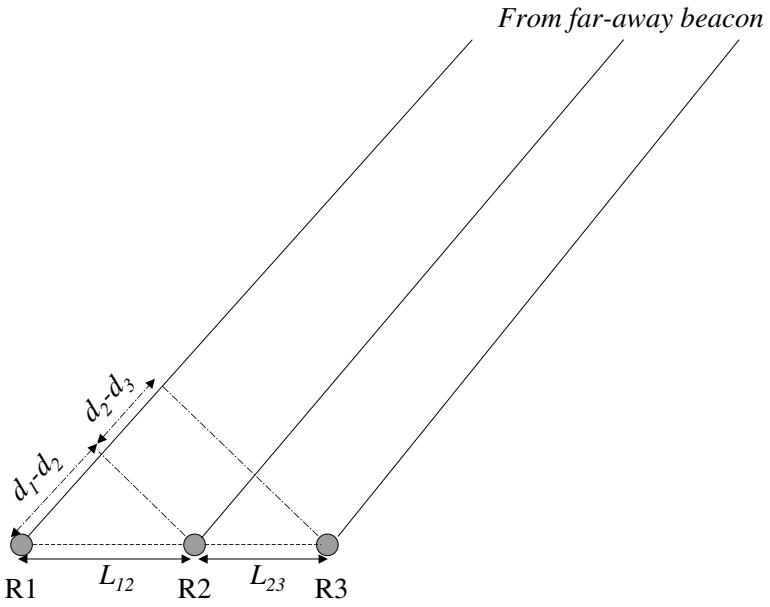


Figure 5-20: Using three receivers to measure $(d_1 - d_2)$.

Subtracting Equation (5.7) from Equation (5.6) and rearranging, we get:

$$L_{12}(n'_{23} - n''_{23}) = L_{23}(n'_{12} - n''_{12}). \tag{5.8}$$

Let us express L_{ij} as $l_{ij}\lambda/2$, which expresses the separation between receivers as an integral multiple of $\lambda/2$. Equation (5.8) is then equivalent to:

$$l_{12}(n'_{23} - n''_{23}) = l_{23}(n'_{12} - n''_{12}), \tag{5.9}$$

where each of the l_{ij} and n_{ij} are integers.

Notice that $|n_{ij}|\lambda \leq \delta d$, the separation in distance between the carrier waveforms at receiver i and receiver j, and $\delta d \leq L_{ij} = l_{ij}\lambda/2$, for each pair $(i, j) = (1, 2), (2, 3)$. This means that $|(n'_{ij} - n''_{ij})\lambda| < 2L_{ij} = l_{ij}\lambda$. (In fact, $|(n'_{ij} - n''_{ij})\lambda|$ may be *equal to* $2L_{ij}$, but only if the beacon lies on the same horizontal plane as the compass. This situation is unlikely in practice, and detectable if it does occur.) Therefore, $|n'_{ij} - n''_{ij}| < l_{ij}$. Thus, if Equation (5.9) is to be satisfied, l_{12} and l_{23} *cannot* be relatively prime.

Hence, it is possible to unambiguously derive an actual phase difference (ϕ_{ij}) in the range of $[0, L_{ij}]$ from an observed one (α_{ij}) by picking L_{12} and L_{23} to be relatively prime integral multiples of $\lambda/2$. For example, we can pick $L_{12} = 2\lambda$ and $L_{23} = 1.5\lambda$. Thus, knowing ϕ, we get the exact δd needed for estimating θ in Equation (5.5).

Consider an array of three ultrasonic receivers with $L_{12} = 2\lambda$, and $L_{23} = 1.5\lambda$. Figure 5-21, plots the variation of observed phase difference $\delta d'$ as a function of the actual distance difference δd for the two pairs of receivers. One of the curves (the

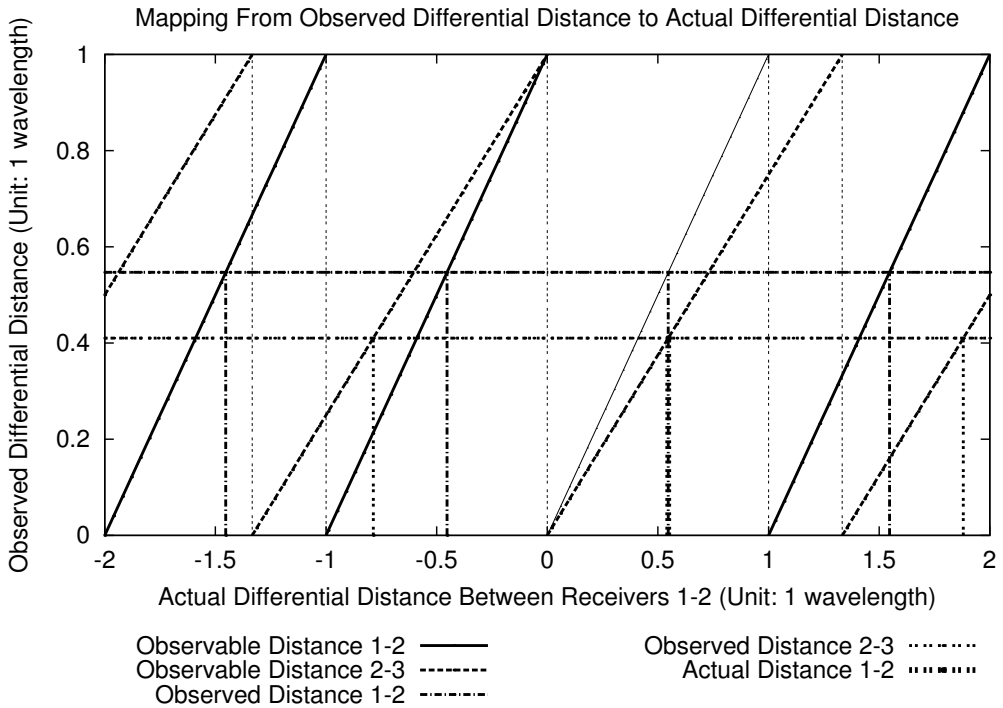


Figure 5-21: Finding the actual differential distance between R_1 and R_2 by using the observed differential distances from (R_1, R_2) and (R_2, R_3).

solid line segments) shows the $\delta d'_{12}$ variation for the receiver pair (R_1, R_2), which are separated by a distance $L_{12} = 2\lambda$. The other curve (the dashed line segments) shows the variation $\delta d'_{23}$ for the receiver pair (R_2, R_3) separated by $L_{23} = 1.5\lambda$. We normalized the curves to show the observed variations of $\delta d'_{12}$ and $\delta d'_{23}$ as a function of δd_{12}; i.e., $\delta d'_{12}$ varies in the range $[0, \lambda]$ as δd_{12} varies in $[-2\lambda, 2\lambda]$.

Each curve is periodic with discontinuities. The observed value $\delta d'$ varies in the range $[0, \lambda]$ because that is the range of measurable distance between two (time-shifted) waveforms whose starting times are not known. The discontinuities are due to the fact that the observable differential distances follow the periodicity of the observed phase differences. The actual differential distances vary in the range $[-L_{12}, L_{12}]$ for δd_{12}, and in the range $[-L_{23}, L_{23}]$ for δd_{23}. But because we have normalized the curves as a function of δd_{12}, the observed phase differential curve for the receiver pair (R_2, R_3) shown in Figure 5-21 also varies in the range $[-L_{23} \cdot L_{12}/L_{23}, L_{23} \cdot L_{12}/L_{23}] = [-L_{12}, L_{12}]$ in the plot. The slope of each line segment is proportional to the normalized separation distance for that pair of receivers. Hence, the normalized curve for (R_1, R_2) has a slope of 1, while the curve for (R_2, R_3) has a slope of $L_{23}/L_{12} = 3/4$.

Note that because L_{12} and L_{23} are relatively prime multiples of $\lambda/2$, the periods (and discontinuities) for the two curves always differ, and the cycles of the two curves (i.e., the discontinuities) do not overlap each other more than once. Consequently, the two curves do not have a repeating pattern within the range of interest, $[-L_{12}, L_{12}]$.

Hence, we get a unique solution for the actual δd value for *any* given pair of observed $\delta d'_{12}$ and $\delta d'_{23}$ values.

From Figure 5-21, we see that any observed value within the range $[0, \lambda]$ can be mapped to four possible solutions for the actual δd_{12}. Let $A^{\delta d'_{12}}$ be the set of possible solutions derived from the observed value $\delta d'_{12}$. Graphically, these are the values on the horizontal axis extrapolated from the four intersections between the $y = \delta d'_{12}$ line and the observable differential distance curve for the receiver (R_1, R_2). Then, given an observed $\delta d'_{12}$, our task is to identify the actual differential distance from the set $A^{\delta d'_{12}}$.

We can use the observed $\delta d'_{23}$ to help us identify the correct solution as follows. From Figure 5-21, the observed $\delta d'_{23}$ can be mapped to three possible solutions for the actual δd_{12}. Again, let $A^{\delta d'_{23}}$ be the set of possible solutions using the observed value $\delta d'_{23}$. Because we are guaranteed a unique solution for any given pair of observed values $\delta d'_{12}$ and $\delta d'_{23}$, we will find exactly one matching solution that exists in both $A^{\delta d'_{12}}$ and $A^{\delta d'_{23}}$. Thus, the final answer for the actual differential distance δd_{12} is a if, and only if, $a \in A^{\delta d'_{12}}$ and $a \in A^{\delta d'_{23}}$.

For example, Figure 5-21 shows that for the observed $\delta d'_{12} = 0.547$ and $\delta d'_{23} = 0.41025$, $A^{\delta d'_{12}} = \{-1.453, -0.453, 0.547, 1.547\}$ and $A^{\delta d'_{23}} = \{-0.786, 0.547, 1.880\}$. Hence, the final solution is $\delta d_{12} = 0.547$ because this value exists in both $A^{\delta d'_{12}}$ and $A^{\delta d'_{23}}$.

One caveat about this algorithm for finding the actual phase differential distance is that measurement errors may produce no matching solution that exists in both $A^{\delta d'_{12}}$ and an $A^{\delta d'_{23}}$. In such a situation, we find the closest matching solution by choosing an $a_{12} \in A^{\delta d'_{12}}$ and $a_{23} \in A^{\delta d'_{23}}$ such that $|a_{12} - a_{23}|$ is minimum. Then, we report the actual differential distance to be $\frac{a_{12}+a_{23}}{2}$.

The biggest disadvantage of this approach is the need for the ultrasonic receivers to be mounted in a straight line with the specified separation between them. Because of the difficulties in mounting the receivers accurately and the manufacturing defects of the receivers, it may be difficult to accurately place the array of receivers as required. One possible solution to this problem is to manufacture the receiver array as a single unit with the required positioning tolerances being maintained during the manufacturing process. Although calibrating the receiver array for possible misalignment may be feasible, the complications due to possible offsets in all three dimensions makes it difficult to devise an efficient calibration scheme. Next, we discuss a solution that we believe is more robust and amenable to mass production.

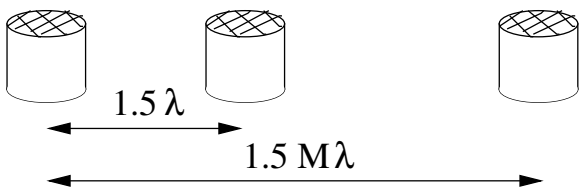


Figure 5-22: Array of three receivers used to solve the phase ambiguity, using a combination of phase and distance difference measurements.

5.3.4 Phase Disambiguation from Phase and Distance Difference

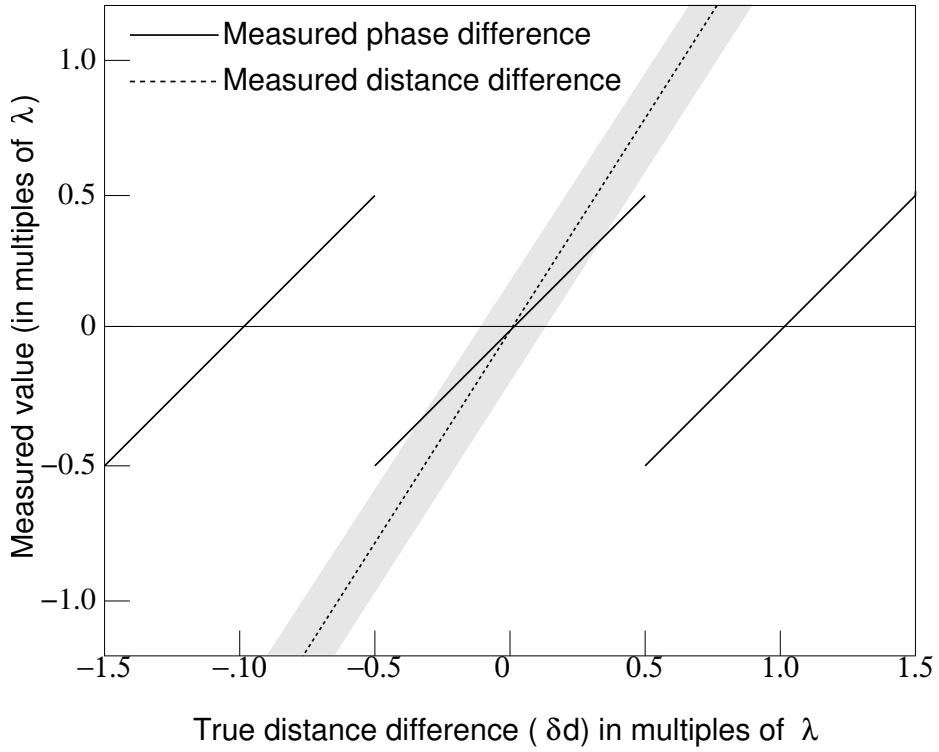


Figure 5-23: The variation of the measured phase difference between the receivers R_1 and R_2, and the measured distance difference between the receivers R_1 and R_3, as a function of the actual distance difference between the receivers R_1 and R_2. The grey band represents the possible values for the measured distance between R_1 and R_3 due to possible measurement errors.

The following scheme avoids the need for highly accurate ultrasonic receiver placement issue of the previous approach by using only a single phase difference measurement between a pair of receivers. We use the differential distance measurement between another pair of receivers to resolve the phase ambiguity. Consider the receiver array shown in Figure 5-22. The distance $L_{12} = 1.5\lambda$ and the distance $L_{13} = M \times 1.5\lambda$ for some M; the three receivers R_1, R_2, and R_3 are at distances d_1, d_2, and d_3 from some beacon B. Let $\delta d_{12} = d_1 - d_2$ and $\delta d_{13} = d_1 - d_3$. Assuming the phase difference-based distance measurement has zero measurement error, the solid line on Figure 5-23 plots the observed distance difference, $\delta d'_{12}$, between the receivers R_1 and R_2, as a function of δd_{12}. Since $L_{12} > \frac{\lambda}{2}$, similar Figure 5-21, $\delta d'_{12}$ vs δd_{12} plot has discontinuities caused by phase ambiguity. The broken line on Figure 5-23 plots δd_{13} vs δd_{12}; since $L_{13} = M \times L_{12}$, the gradient of the broken line is M. Let $\delta d'_{13}$ denote the measured differential distance between R_1 and R_3; if differential distance estimation has a maximum error of ϵ, then $\delta d_{13} - \epsilon \leq \delta d'_{13} \leq \delta d_{13} + \epsilon$. The shaded area in Figure 5-23 shows the set of possible values of $\delta d'_{13}$ vs d_{12}.

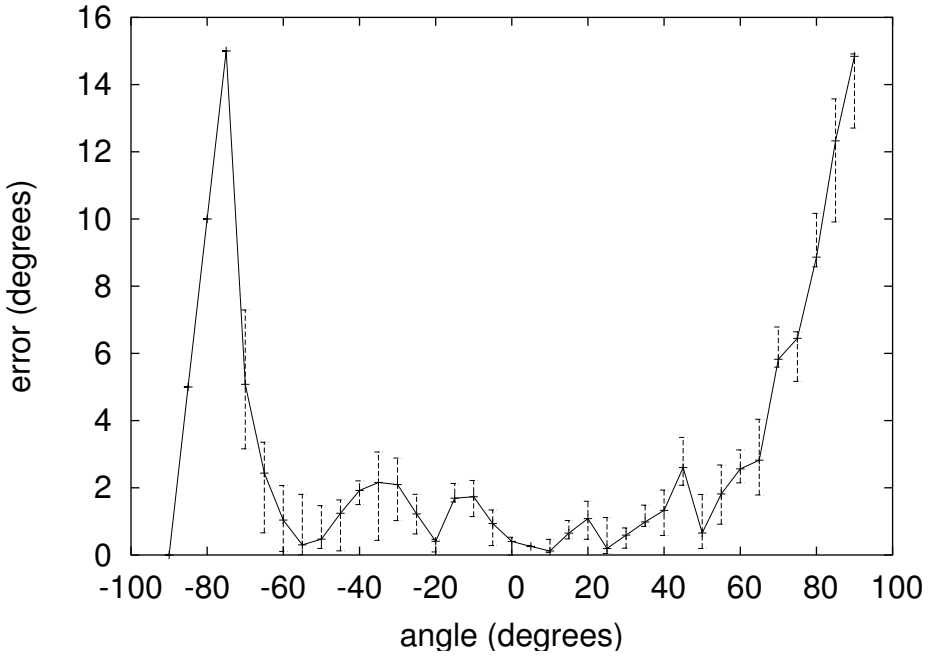


Figure 5-24: The orientation measurement error of Cricket as function of the angle between the perpendicular to the receiver array and a beacon.

For some measured value of $\delta d'_{12}$, there are three possible values, $p < q < r$, of δd_{12}. We observe that $r = q + \lambda$ and $q = p + \lambda$. The corresponding δd_{13} values are (Mp, Mq, Mr). Since $r = q + \lambda$ and $q = p + \lambda$, any pair of these δd_{13} values are separated by at least $M\lambda$. If $M\lambda > 2\epsilon$, then we can distinguish between the different values of δd_{13} (and δd_{12}) using the measured values $\delta d'_{13}$. Hence, if $M\lambda > 2\epsilon$, then for a given value of $\delta d'_{13}$, we can use $\delta d'_{13}$ to determine a unique value for δd_{12}.

For the 40 kHz ultrasonic signal used in Cricket, $\lambda > 8$ mm. From 5-17, the differential distance estimation error, ϵ, is less than 10 mm. Hence, we need to select M such that, $M \geq 20/8 = 2.5$. Hence, in Cricket, we can resolve the phase ambiguity by selecting $L_{13} = 2.5L_{12} = 3.75\lambda$.

We used the following set up to determine the phase and distance difference-based phase disambiguation performance and the orientation measurement performance of Cricket. We placed three receivers as shown in Figure 5-22 with $L_{12} = 1.5\lambda$ and $M = 2.7$. We placed the array of three receivers on a rotary table, and we placed a beacon 3 m away from the center of the rotary table on the horizontal plane containing the array. We measured the distance difference $\delta d'_{12}$ using phase difference, and the distance difference $\delta d'_{13}$ using ultrasonic signal arrival times. We collected data by rotating the rotary table in the range $(-90^o, 90^o)$ in increments of 5^o. For each angle, we collected 50 samples of each measurement.

Using the phase and distance difference-based approach, we manged to correctly resolve the phase ambiguity 100 % of the time. Figure 5-24 shows the magnitude of the average angle measurement error for different values of rotation, the error bars represent the minimum and the maximum error. We observe that within the

$(-45^o, 45^o)$ range, the average orientation measurement error is less than 3^o.

5.4 Obtaining Listener Orientation using Receiver Arrays

The previous section described how to obtain the distance difference δd of the two ends of an array of ultrasonic receivers accurately. We can substitute this value in Equation(5.5) to obtain the angle θ between the beacon and the perpendicular to the receiver array. However, the accuracy of θ depends on both the accuracy of δd and the value of θ itself. We can analyze how the value θ affects the accuracy of determining θ as follows. From Equation(5.5),

$$cos\theta \times d\theta = \frac{d(\delta d)}{L}. \tag{5.10}$$

According to Equation (5.10), θ is least sensitive to changes in δd when $\theta = 0$ where the error in δd produces a similar error in θ; θ is most sensitive to δd when $|\theta| = \pi/2$, where a small error in δd could produce an unbounded error in θ. We define the range $[-\frac{\pi}{4}, \frac{\pi}{4}]$ as the *usable* range of θ. Within this range, the measurement error of δd is magnified by at most $\sqrt{2}$. We can visualize the region containing beacon locations that result in usable θ values in Figure 5-25; this *usable region* is the 3D shape that results from subtracting two cones from 3D space.

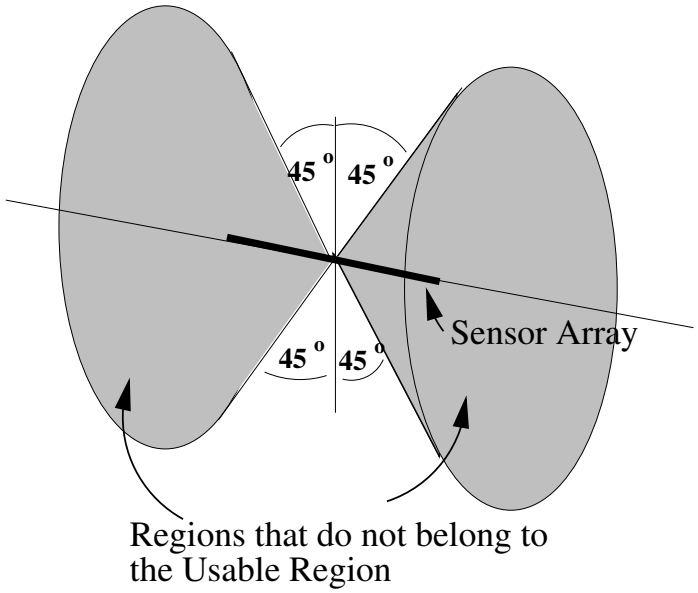


Figure 5-25: The shaded conical regions represent the beacon locations where the angle, θ, between a beacon and the normal to the receiver array is $> 45^o$. The *usable region* is the space outside the shaded region.

Suppose we know the position, P_b, of of some beacon B, which is located at an angle θ to the perpendicular to ultrasonic receiver array, and the position P_r of one end

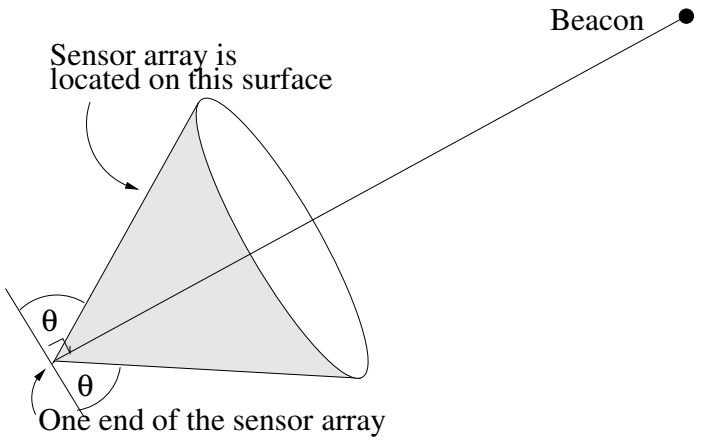


Figure 5-26: Once the angle θ between the beacon and the perpendicular to the receiver array, and the position of the end of the receiver array is known, the receiver array is confined to the surface of a cone.

of the receiver array The receiver array should be located on the surface of the cone with the apex P_r with principal axis (P_r, P_b), and the vertex angle $(\pi 2\theta)$ (Figure 5-26). If we have the coordinate and angle information with respect to a second beacon that is not co-linear with the (P_r, P_b) line, then, the receiver array must lie on either of the two lines along which the two cones intersect (Figure 5-27). Now, if we have coordinate and angle information for a third beacon, we can determine the location of the receiver array uniquely as the intersection of the cone from the third beacon and one of the previous two lines provided the three beacons and the point P_r do not lie on the same plane (otherwise, all three cones intersect along the same two lines). We next examine different techniques for obtaining the listener orientation from the orientation of one or more receiver arrays to multiple beacons.

5.4.1 Orientation from Three Beacons and Two Receiver Arrays

As we described earlier, we can determine the orientation of a receiver array uniquely by measuring angles to three beacons if the three beacons and the receiver array do not lie on the same plane (we also need the coordinates of one point on the receiver array) . In this setup, only two of the three beacons may need to be within the usable region. We use these two beacons to accurately compute two possible orientations for the receiver array; if these two orientations have a large enough separation, we can use a less accurate angle measurement from a third beacon to disambiguate between these two lines (if the two possible directions are close to each other, then all three beacons must lie within the usable region). We can use two receiver arrays that are not parallel to each other to uniquely determine the listener orientation. Figure 5-28 and Figure 5-29 show a possible configuration of five receivers resulting in two orthogonal receiver arrays of three receivers each (one receiver is common to both arrays).

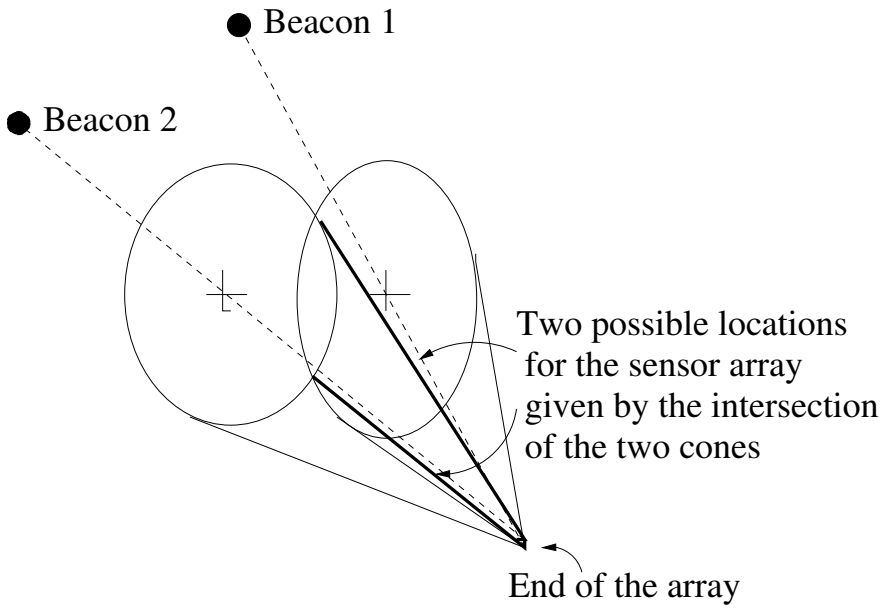


Figure 5-27: Using the angles θ and β measured with respect to two beacons, and the position of the one end of the receiver array, we can obtain two possible locations for the receiver array—obtained from the intersection of two cones.

5.4.2 Orientation from Two Beacons and an Inclinometer

We can use an inclinometer to reduce the number of beacons required for determining the listener orientation. For example, we can attach a 2D inclinometer such as ADXL213 from Analog Devices to the plane containing the two receiver arrays in Figure 5-28. The inclinometer enables us to measure the tilt of the each receiver array with respect to the horizontal plane. If one of the receiver arrays is inclined to the horizontal by an angle α, as Figure 5-30 shows, that receiver array should lie on a cone with its axis perpendicular to the horizontal plane, with one end of the receiver array as its vertex (we obtain the coordinates of the vertex using cricket position

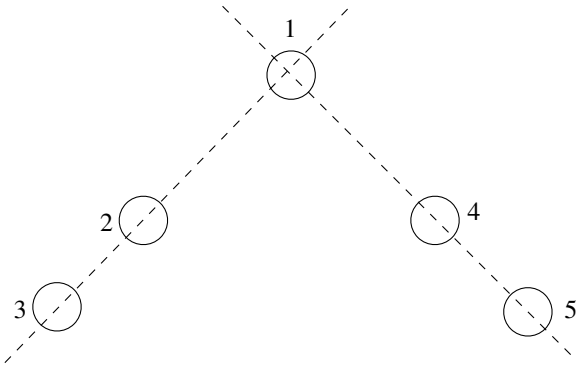


Figure 5-28: Five receivers arranged in to two orthogonal receiver arrays of three receivers each (one receiver is shared by both arrays).

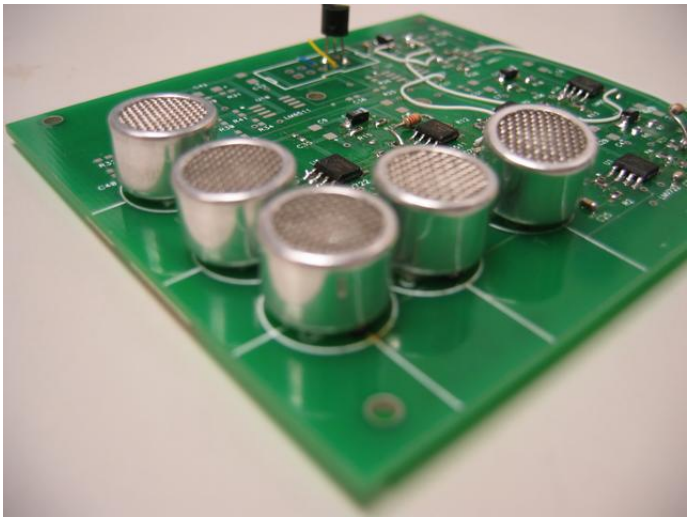


Figure 5-29: A Cricket compass board with five receivers arranged in two arrays of three receivers each.

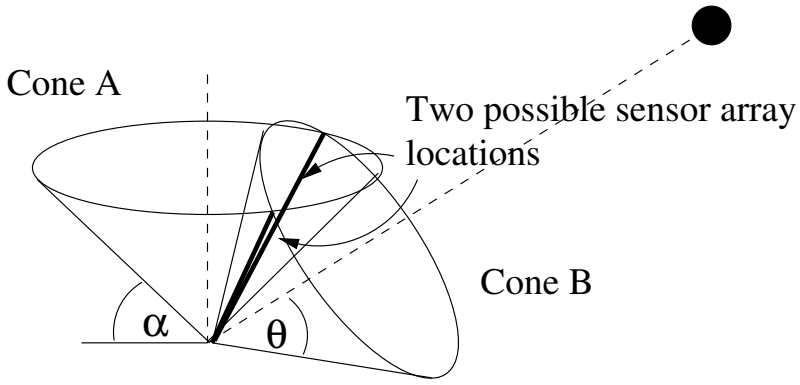


Figure 5-30: The angle α from the inclinometer and the angle β from the beacon results in two possible locations for the receiver array.

estimation algorithms). Assuming that we can measure α with high accuracy, we can determine two possible solutions to the orientation of the receiver array using the angle θ measured with respect to a beacon in the usable region. We can use a second beacon to disambiguate between these two possible solutions.

Once the orientation of the first receiver array has been uniquely determined, the second receiver array is located on a disk perpendicular to the first receiver array, as shown in Figure 5-31. Since we know the inclination β of the second receiver array to the horizontal plane (from the inclinometer), we can obtain two possible solutions for the orientation of the second array. We can disambiguate between these two points using the inclination of the second receiver array to some beacon.

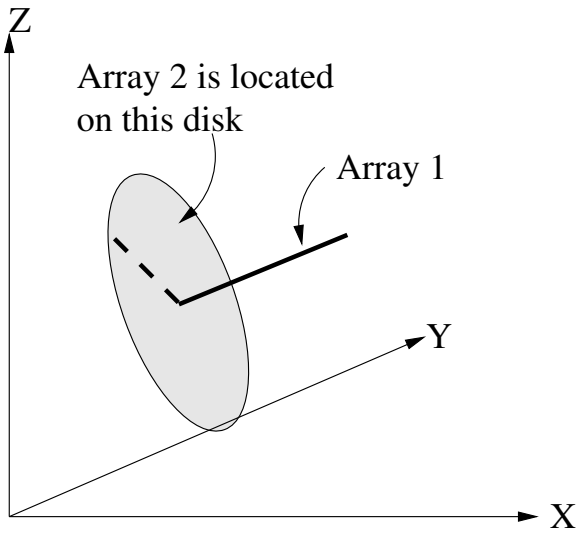


Figure 5-31: Once the orientation of the first array is known, the second array lies on a disk perpendicular to the first array.

5.4.3 Orientation of a Horizontal Listener

There are certain situations, such as when mounted on a robot, where the listener always stays parallel to the horizontal plane. If the X-Y plane of the Cricket coordinate assignment is also parallel to the horizontal plane, we can simplify the listener orientation computation as follows.

Consider Figure 5-32, where the listener is parallel to the horizontal plane, and the receiver array on the listener is at an angle θ to the direction of the beacon on the horizontal plane (the listener is at an angle θ to the projection of the line joining the beacon and the listener on the horizontal plane). Figure 5-33 shows the beacon B from Figure 5-32 projected on to the horizontal plane containing the listener. In this figure, x_1 and x_2 are the projections of distances d_1 and d_2 on to the horizontal plane.

From Figure 5-32:

$$x_1^2 = d_1^2 - z^2 \tag{5.11}$$

$$x_2^2 = d_2^2 - z^2 \tag{5.12}$$

$$x = \sqrt{\bar{d}^2 - z^2}$$

where $\bar{d} \approx \frac{d_1+d_2}{2}$ when $d_1, d_2 \gg L$.

From Figure 5-33:

$$x_1^2 = (\frac{L}{2}\cos\theta)^2 + (x - \frac{L}{2}\sin\theta)^2$$

and

$$x_2^2 = (\frac{L}{2}\cos\theta)^2 + (x + \frac{L}{2}\sin\theta)^2$$

$$\Rightarrow x_2^2 - x_1^2 = 2LyX\sin\theta.$$

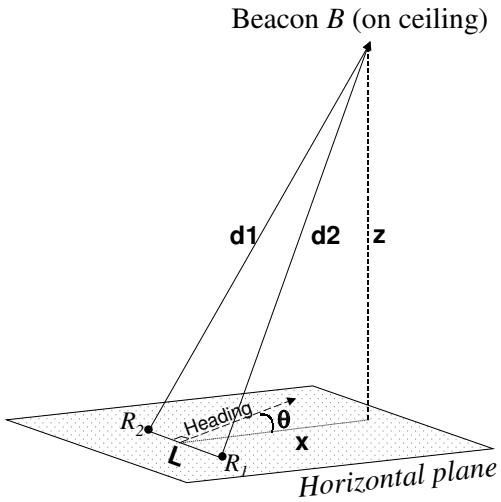


Figure 5-32: Determining the orientation of a listener held parallel to the horizontal plane. Observe that θ is the angle between the receiver array and the projection of the line joining the beacon and the receiver array, on the horizontal plane.

Substituting for x_1^2 and x_2^2 from Equations (5.11) and (5.12), we get:

$$\sin\theta = \frac{d_2 + d_1}{2Lx} \cdot (d_2 - d_1). \tag{5.13}$$

This equation may be rewritten as:

$$\sin\theta = \frac{d_2 - d_1}{L\sqrt{1 - (\frac{z}{\bar{d}})^2}} \tag{5.14}$$

Equation 5.14 implies that we can obtain the orientation from the distance difference $\delta d = (d_2 - d_1)$, the z coordinate of the beacon, and the distance to the beacon from the receiver array ($\bar{d}$). However, as we mentioned earlier, we limit the θ to the range $[-\frac{\pi}{4}, \frac{\pi}{4}]$ to reduce errors; we use five receivers arranged as two arrays of three receivers each, to determine the orientation (Figure 5-28). For a given beacon, we use the receiver array that results in an angle $|\theta| \leq \frac{\pi}{4}$. For a given receiver array, there are two possible solutions for the beacon direction due to symmetry (Figure 5-34); as shown in Figure 5-35, we use the other receiver array to break symmetry and uniquely determine the beacon direction.

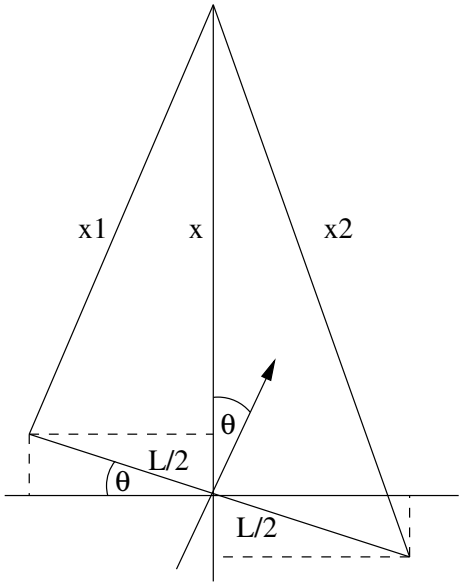


Figure 5-33: This figure is the projection of the beacon onto the horizontal plane containing the listener.

5.5 Chapter Summary

This chapter examined how Cricket listeners determine their space, position, and orientation using distance measurements to nearby beacons. This chapter first examined how to determine space by deploying beacons to demarcate boundaries between spaces. Next, this chapter described how to obtain listener position using distance samples and known beacon coordinates. This chapter also examined how to obtain accurate differential distance estimates to compute listener orientation, and also described an ultrasonic sensor placement method to resolve phase ambiguity in phase difference-based distance estimation. Finally, this chapter also described multiple sensor array arrangements to obtain listener orientation. The next chapter describes how to use distance samples collected at a mobile listener to obtain inter-beacon distances and build a rigid graph of beacons for computing beacon coordinates.

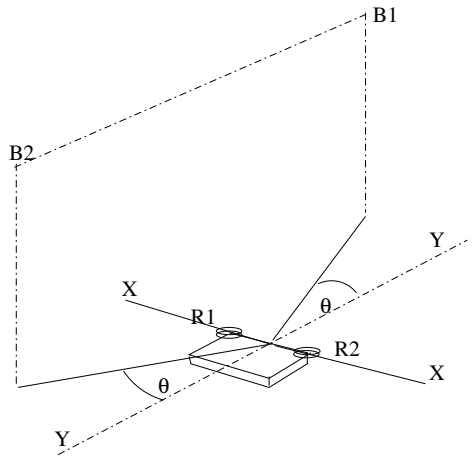


Figure 5-34: θ is ambiguous—the beacon can be at either $B1$ or $B2$.

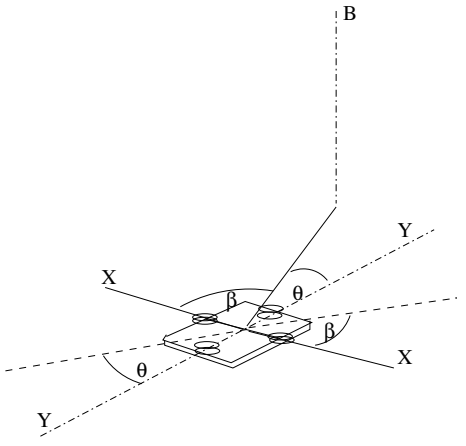


Figure 5-35: One receiver array computes the angle, the other receiver array breaks the symmetry of the beacon position.

Chapter 6

Mobile-assisted Topology Generation for Beacon Localization

Chapter 5 described how Cricket listeners obtain different types of location information from the distances to near-by beacons and the beacon coordinates. Chapter 7 describes a localization algorithm that computes beacon coordinates using inter-beacon distances. However, as Chapter 4 describes, the beacons cannot directly measure inter-beacon distances due to two reasons; first, the weak ultrasonic signal strength along a plane containing beacons prevent coplanar beacon from measuring inter-beacon distances, and second, the line-of-sight requirement for ultrasound-based ranging prevent the beacons from obtaining inter-beacon distances when the beacons are deployed across an entire floor of a building. We note that, although it might be possible to overcome the lack of inter-beacon distances due to weak ultrasonic signal strength by using better sensor, the lack of line-of-sight is a fundamental problem in indoor environments when the ranging system requires line-of-sight for distance measurements.

Moreover, when computing beacon coordinates using inter-beacon distances, it is important that there are enough inter-beacon distances such that the set of distances uniquely define how the beacons are located with respect to each other. For example, consider the three beacons shown in Figure 6-1. If we compute coordinates using

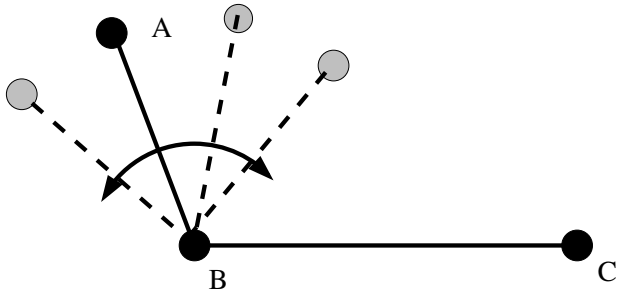


Figure 6-1: For given three beacons, two pair-wise distances do not uniquely define the relative coordinates of the beacons because the two distances do not uniquely fix the angle between those two edges.

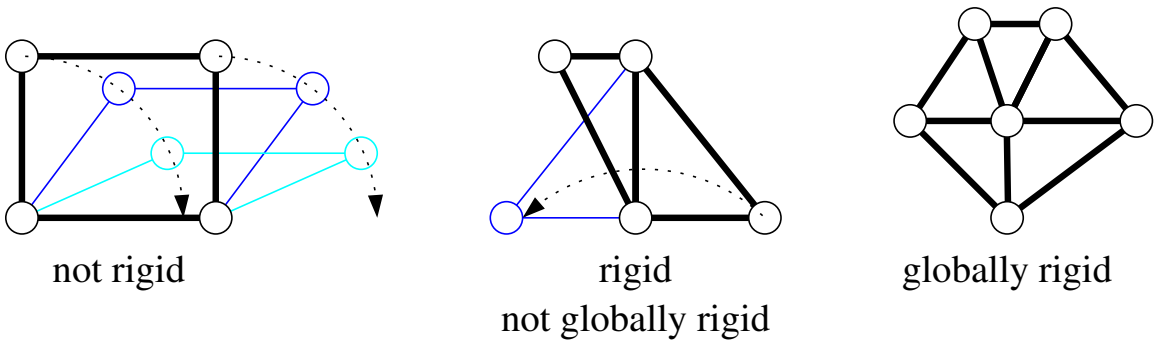


Figure 6-2: Examples of graphs that are not rigid (flexible as a bar-and-joint framework), rigid but not globally rigid (multiple embeddings), and globally rigid (one embedding up to rotation, translation, and reflection).

only two distances, the resulting coordinates can be different from the actual beacon deployment; only a coordinate assignment that uses all *three* inter-beacon distances is guaranteed to be consistent with how beacons are located with respect to each other. Hence, before using a set of inter-beacon distances to compute a beacon coordinate assignment, it is necessary to make sure that there are enough inter-beacon distances such that the resulting coordinate assignment reflects the actual beacon deployment.

This chapter discuss Mobile-Assisted Topology generation (MAT), an algorithm which enables us to obtain inter-beacon distances using distance measurements taken at a listener [73]. It also describes how to collect enough inter-beacon distances such that a localization algorithm can produce a valid coordinate assignment that reflects the true relative positioning of the beacons. Section 6.1 describes how the *rigidity* of the graph consisting of the beacons and the set of inter-beacon distances determines if there are enough inter-beacon distances to compute a valid coordinate assignment. Section 6.2 describes the benefits of a receiver-based approach to obtain inter-node distances when solving the node localization problem in general, in both indoor and outdoor environments. Next, Section 6.3 describes how to obtain inter-beacon distances from distances collected at multiple listener positions under different beacon configurations. Section 6.4 describes how to explore the environment and obtain enough inter-beacon distances for computing a valid beacon coordinate assignment. Section 6.5 uses simulations to examine how the accuracy of inter-beacon distance computations depend on the number of listener positions and the layout of the listener positions. Section 6.6 presents results from running the described algorithms on a typical indoor deployment of 21 beacons.

6.1 Rigidity Requirements for Beacon Localization

Consider a physical embodiment of a graph, consisting of linkages representing the edges on the graphs and flexible joints representing the vertices. The number of different physical shapes this embodiment can take, based on how much we can flex it or flip parts of it, determine the degree of rigidity of this graph.

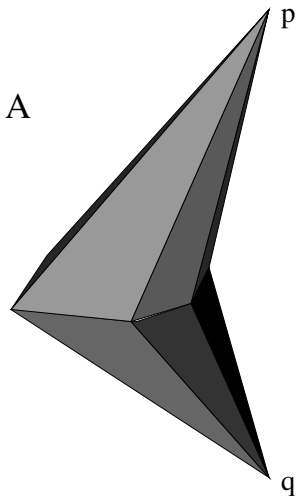


Figure 6-3: A rigid 3D structure that satisfies the two properties described by Hendrickson.

Rigidity theory identifies three classes of graphs[20, 40, 39] (Figure 6-2). A graph that can be flexed while maintaining the edge lengths is called *not rigid*. Because flexing results in infinitely many different relative node positions, we cannot obtain a unique relative node layout from the edge lengths of a non-rigid graph. A graph that cannot be flexed, but which may be subject to "local flips", while maintaining the edge lengths, is called *locally rigid*; because a locally rigid graph has a finite number of different relative node arrangements that satisfy the set of inter-node distances, we cannot obtain a unique node layout from the edge lengths of a locally rigid graph. A graph where the set of edge lengths uniquely determine how nodes are located with respect to each other is called *globally rigid*; hence, for a node localization algorithm to produce a coordinate assignment that reflects the actual beacon deployment, we need a set of edge lengths that results in a rigid graph of beacons.

Hendrickson introduced global rigidity as as an important variation on the well-studied concept of local rigidity [20, 40, 39]. Hendrickson showed that, for a graph to be generically globally rigid in d dimensions, it must satisfy two properties: (1) the removal of any d vertices must leave the graph connected ($(d+1)$-connectivity), and (2) the removal of any edge must leave the graph generically locally rigid. Both these properties can be checked in polynomial time. Connelly [19] proved that these two properties are insufficient in 3D: they do not imply generic global rigidity of a 3D graph. For example, consider the structure shown in Figure 6-3, assume that this is globally rigid and satisfies the two properties mentioned above. Next, consider the 3D structure shown in Figure 6-4, obtained by connecting two instances of the structure in Figure 6-3 at two points p and q. Since the two substructures satisfy property (1), the resulting structure also satisfies the property (1). The resulting structure is only locally rigid, since each substructure can be rotated about the points p and q, according to property (2), removing any edge will make one substructure locally rigid; hence, removing a single edge on either of the substructures makes the resulting structure locally rigid. Hence, the resulting structure satisfies both properties (1) and

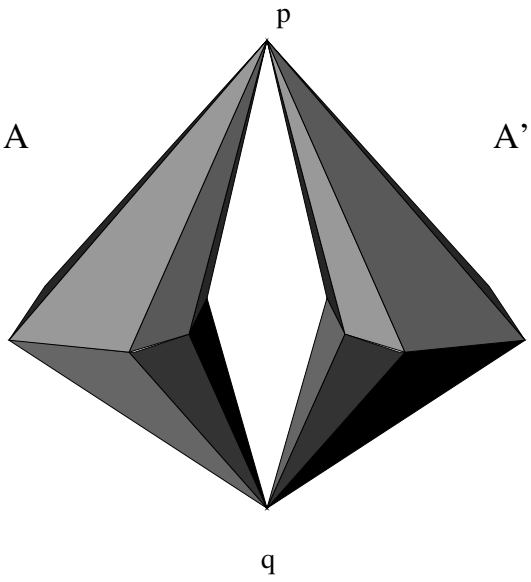


Figure 6-4: A 3D structure obtained by connecting two instances of the structure in Figure 6-3. This structure is only locally rigid, since each substructure can be rotated about $p\ q$ with respect to the other substructure.

(2). Since each substructure can be rotated about the line $p\ q$ with respect to the other substructure, the resulting structure is only locally rigid. Hendrickson conjectured that above two properties exactly characterize generic global rigidity in 2D, and this conjecture was recently proved by Jackson and Jordán [52].

Thus, global rigidity is well-understood in 2D, but not in 3D. Since we use a mobile listener to explore the environment and compute inter-beacon distances, we need not test if a given graph is rigid, rather we *construct* graph structures that are guaranteed to be globally rigid and therefore localizable in both 2D and 3D. Section 6.4 presents an algorithm which guides a mobile user or a robot carrying a listener to explore the environment and build a rigid graph.

6.2 The Need for MAT

Although we are specifically interested in obtaining inter-beacon distances within the Cricket system, there are other situations in general where inter-node distances are used to compute node coordinates. For example, any indoor location system that uses a collection of nodes as reference points needs to determine the position of these nodes within some coordinate system; unless all these nodes can determine their positions independently, say using GPS receivers, these nodes needs to run a localization algorithm, most probably based on inter-node distances, to compute their coordinates. Node localization is an important topic in sensor networks, in both indoor and outdoor environments; location information in sensor networks enable location aware sensing and geographic routing of messages. Mobile-assisted topology generation solves three important practical problems that have made it hard to deploy

previously proposed localization algorithms in many real-world systems.

6.2.1 Lack of Inter-Node Distances

A collection of deployed nodes may not be able to directly measure the inter-nodes distances for several reasons.

The lack of line-of-sight connectivity may prevent the nodes from obtaining direct node-to-node distances. Most of the ranging technologies used for accurate indoor ranging today, including time-of-flight of ultrasound, laser, and infrared, require line-of-sight between the transmitter and the receiver. Even technologies that do not need line of sight, such as ultrawideband (UWB) radio, have better accuracy when line-of-sight connectivity is available. As we discussed in Chapter 4, Cricket uses time-of-flight of ultrasonic to measure distances and it is almost impossible to deploy beacons in an indoor environment to achieve a reasonable line-of-sight connectivity across the beacons. For example, it is hard to obtain distances between beacons placed inside and outside a room in a standard building. Another motivating example is a collection of sensor nodes with ranging capability that are deployed in an outdoor environment where buildings and trees may prevent direct distance measurement due to lack of line-of-sight; however, an aircraft flying over these nodes may have enough visibility to collect distances samples to compute inter-node distances.

The lack of omni-directional ranging may prevent reference nodes in a location system from obtaining pairwise distances. Because the primary goal of a location system is to help mobile devices obtain distance and location information, a key requirement is to provide maximum coverage to users. As a result, directional ranging transmitters on reference nodes are usually pointed toward where the users are likely to be, rather than toward other reference nodes. For example, due to the radiation pattern of the ultrasonic transducers used, Cricket has a 12 m range when the transmitter and the receiver are facing each other but only a <2 m mutual range when they are on the same horizontal plane facing away from the plane (e.g. downwards from a ceiling). Building omni-directional ranging is usually more expensive (e.g. it requires multiple transceivers) and entails hardware changes; furthermore, it seems wasteful because localization is not a continually running process.

6.2.2 Sparse Node Deployments

Although a dense node deployment of reference nodes helps achieve good coverage in a location system, economic considerations often force sparse node deployments. Sparse deployments reduce inter-node connectivity and could lead to a structure that is not rigid. For example, a room with four reference nodes where only four distances (forming a quadrilateral among the nodes) are known leads to a non-rigid structure. In such cases, a localization algorithm cannot find a valid coordinate assignment for the nodes, because there are too few known constraints.

Mobile-assisted topology generation is well-suited to collect enough distance samples such that the resulting inter-node distances form a rigid structure. Because the reference nodes in a location system are deployed to cover an area where users move, a

moving user or robot can take advantage of the many positions from where distances to the nodes can be obtained.

6.2.3 Geometric Dilution of Precision (GDOP)

The distance measurements used to compute node coordinates always have some error. These measurement errors get reflected as errors in the computed node coordinates. The magnitude of the final error depends on both the magnitude of the measurement error and the geometry of the structure induced by the nodes and edges. The contribution due to geometry is called the *geometric dilution of precision (GDOP)* [95][1], and is defined as the ratio between the computed coordinate error and the measurement error. GDOP represents the factor by which the distance measurement error gets multiplied when it is used to compute node coordinates. When distance measurements are used for computing node coordinates by solving for an exactly constrained system of equations, we get GDOP > 1.

It is well-known that using an over-constrained system of equations tends to reduce GDOP errors [83]. Adding additional constraints in conventional approaches leads to increased node density; in contrast, with mobile-assisted topology generation, a mobile unit can move around a region and usually obtain as many additional constraints as are necessary. Since obtaining additional constraints is not cumbersome, mobile-assisted topology generation can improve the accuracy of coordinate estimation by reducing the adverse effects of GDOP.

6.2.4 Other Benefits of MAT

Although the use of mobile device positions as "virtual" nodes to add more constraints to a weakly connected graph and running a localization algorithm on all the nodes seems like a simple extension to standard auto-localization, the flexibility offered by the mobile device acting as a virtual node creates a set of opportunities that makes our approach different from traditional node localization. For instance:

- The dynamic nature of the mobile assisted scheme enables us to evaluate the currently available distance information "on the fly", and navigate the mobile to obtain additional distances as required.
- The additional virtual nodes corresponding to mobile positions do not have the associated cost of additional physical nodes. The only criteria that limit the number of such positions are the associated computational and storage overhead, both of which are usually ample on current handheld devices. Even if they were not ample, it is possible to store information associated with dozens of mobile positions on even impoverished mobile hardware.
- The reference and virtual (mobile) nodes typically occupy different regions in space. In a typical location infrastructure deployment, for example, the reference nodes will be located close to the ceiling of a room and above the space

[1]GDOP is a well-known problem that arises in all location systems, including GPS.

occupied by users. In contrast, the virtual nodes, corresponding to mobile locations, will be located close to the users. This separation can help the localization perform well, as described in Chapter 7.

6.3 Computing Inter-Node Distance using a Mobile Node

This section describes different approaches that we can use to compute inter-node distances using a mobile node. This section also describes the absolutely minimum number of mobile positions required and the added restrictions that need be placed on these positions based on the number of fixed nodes that are within the range of the mobile node.

6.3.1 Calculating the Distance Between Two Nodes

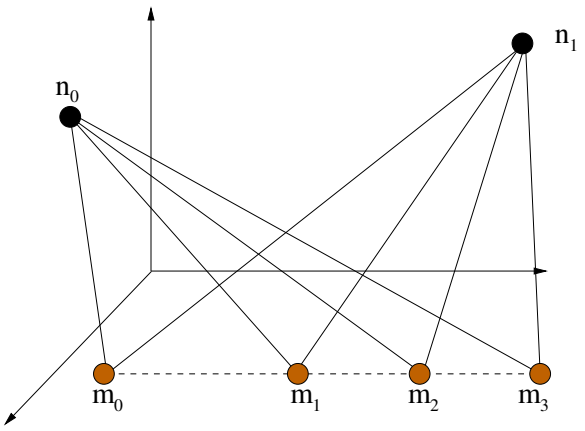


Figure 6-5: Computing the distance between two nodes by measuring distances from four points, where the two nodes and the four points lie on the same plane.

We first examine how to compute the distance between two nodes n_0 and n_1 by measuring distances these nodes from various locations of a mobile node m. Initially, we have a single unknown, the distance $\|n_0 - n_1\|$, and no known information. Now, every time we collect distances from a new location of the mobile node, we introduce two constraints (the two distances), and three new unknowns (the coordinates of the mobile node's location). This approach does not work since we add more unknowns than constraints with each mobile node location. If we assume that the mobile node m stays on a plane, the new constraints only balance the new unknowns, so we cannot determine the original unknown.

One approach is to move the mobile along a line in a plane containing both n_0 and n_1. A practical example of this idea is when the mobile moves along a projection of the line through n_0 and n_1 (but not along the line containing n_0 and n_1) as shown in Figure 6-5. We now measure distances from four mobile locations. The first location

introduces three unknowns and two constraints; after the first point, we have two unknowns. Since the two lines lie on the same plane, the second mobile point must lie on the plane P defined by the two nodes and the first mobile position; hence, the second mobile position introduces three constraints (the two distances and the requirement that it must lie P) and three unknowns. The third mobile position introduces four constraints (the two distances and the requirement that it must lie on the line L defined by previous two mobile points) and three unknowns, resulting in a net of one unknown. Next we obtain distances from the fourth mobile position, which introduces four more constraints and three unknowns. With four mobile positions the number of unknowns equals the number of constraints. These constraints uniquely determine the geometry provided we know that the two stationary nodes lie on the same side with respect to the line joining the four mobile positions (this additional condition is necessary because the distances to the fixed nodes are symmetric about the line joining the mobile positions).

Proposition 6.1. *The geometry of six coplanar points $n_0, n_1, m_0, m_1, m_2, m_3$, where m_0, m_1, m_2, m_3 are collinear, is determined by the distances $\|n_i - m_j\|$ for $i = 0, 1$ and $j = 0, 1, 2, 3$.*

The solution geometry can be obtained using standard optimization techniques. Although the minimum number of mobile locations required for a solution is four, using a larger number locations would reduce the error caused by GDOP. Note also that it may not be practical to constrain movement in this fashion.

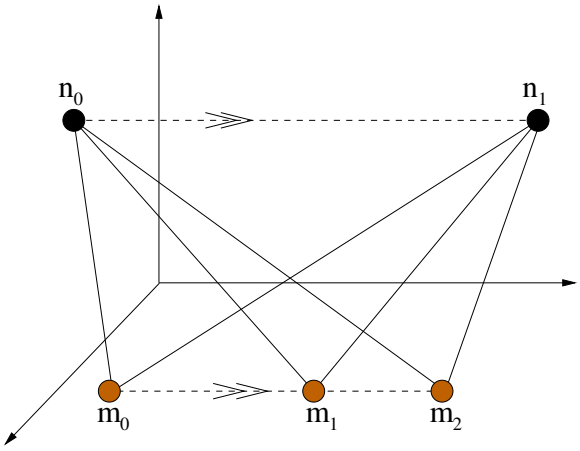


Figure 6-6: Computing the distance between two nodes by measuring distances from three points on a parallel line.

A slightly different approach is to collect distances from multiple mobile positions that lie on a line parallel to the line joining n_0 and n_0. Since the two lines are parallel to each other (in contrast to being on the same plane as in the previous approach) it adds an extra constraint compared to the previous approach, and requires distances from only three mobile positions (Figure 6-6).

A simpler approach when both stationary nodes are at a fixed height from the ground is to position the mobile directly under one of the nodes. Now we can use

the Pythagorean theorem to compute the distance between two nodes as $\sqrt{d_1^2 - d_0^2}$, where d_0 is the distance to the node directly above and d_1 is the distance to the other node from the mobile [61]. Unfortunately, positioning the mobile directly under a node is error-prone; manual placement could cause an error of several centimeters.

6.3.2 Calculating Distances Among Three Nodes

Next we consider how to compute the pairwise distances between three nodes n_0, n_1, and n_2 by measuring distances to these three nodes from various locations of a mobile node m. Initially, there are three unknowns, the distances $\|n_0 - n_1\|$, $\|n_1 - n_2\|$, and $\|n_2 - n_0\|$. If we do not assume any restriction on the placement of mobile positions, for each mobile position we introduce three unknowns and three restrictions, making it impossible to solve for the initial unknowns. Thus we impose one restriction: that the mobile positions all lie on a common plane. This restriction is easy to achieve in practice by moving the mobile receiver at a fixed height from the ground, assuming that the ground is flat. Now if we have k mobile locations, we obtain $k-3$ additional coplanarity constraints. (The first three mobile locations are automatically coplanar, as are all three points in space.) Therefore, $k = 6$ mobile locations are necessary to reduce the number of degrees of freedom to 0. Moreover, these constraints uniquely determine the geometry provided we know that the stationary nodes are all above the plane containing the mobile positions (this additional condition is required to break the symmetry around the plane containing the mobile positions).

Proposition 6.2. *The geometry of three non-collinear points n_0, n_1, n_2 above the plane containing six coplanar points $m_0, m_1, m_2, m_3, m_4, m_5$, no three of which are collinear, is determined by the distances $\|n_i - m_j\|$ for $i = 0, 1, 2$ and $j = 0, 1, 2, 3, 4, 5$.*

As above, the solution geometry can be obtained from standard optimization techniques. Although the minimum number of mobile locations required for a solution is six, a larger number of points reduces the error caused by GDOP.

6.3.3 Calculating Distances Among Four or More Nodes

Here we examine how to compute the pairwise distances between $j \geq 4$ nodes n_1, n_2, ..., n_j by measuring distances to these nodes from various locations of a mobile node m. Now each mobile position adds j (> 3) constraints and only 3 unknowns, for a reduction in the number of degrees of freedom by $j - 3 \geq 1$. Initially there are $3j - 5$ unknowns: three coordinates per stationary node, minus 3 degrees of translational motion and 2 degrees of rotational motion. Thus we require at least $\lceil (3j-5)/(j-3) \rceil$ mobile positions for the number of degrees of freedom to reduce to 0.

It is impractical to assume that j is too large, both because it requires a large node density to have so many line-of-sight paths (especially indoors) and because solving the resulting system of polynomial equations grows in difficulty (though for any fixed j it is polynomial). Therefore we focus on the simplest form of this case, $j = 4$. Then $\lceil (3j-5)/(j-3) \rceil = 7$. Again, we find that 7 mobile positions suffice to uniquely determine the geometry, as the degree-of-freedom analysis predicts:

Proposition 6.3. *The geometry of eleven points* $n_1, n_2, n_3, n_4, m_1, m_2, m_3, m_4, m_5, m_6, m_7$*, no four of* n_i*s are coplanar, is determined by the distances* $\|n_i - m_j\|$ *for* $i = 1, 2, 3, 4$ *and* $j = 1, 2, 3, 4, 5, 6, 7$*.*

The non-coplanarity assumption requires no more than three beacons to be at a constant distance from the floor. This property is easy to arrange on the ceiling by varying-length mounts. If there is no information on the coplanrity of the fixed nodes, it is safer to use the three fixed node approach discussed earlier.

6.4 Building Rigid Graphs

Although testing for global rigidity is difficult, we can *build-up* 3D and 2D structures that are guaranteed to be globally or locally rigid. In mobile-assisted topology generation, we use a mobile receiver to explore a geographic area and incrementally build a rigid graph of nodes by adding edges between the nodes as necessary. The receiver starts by adding enough constraints to nearby nodes to create a cluster of rigid nodes. Then it explores the region for new nodes while adding extra constraints to ensure the rigidity of the whole graph.

The idea of this incremental construction was first used in 2D by Coullard and Lubiw [22] to prove global rigidity of certain 2D visibility structures. It has since been used in several incremental localization algorithms [81, 67].

6.4.1 Creating a Globally Rigid 3D Structure

This section examine how to build up 3D structures that are guaranteed to have global rigidity. We fist show that we can uniquely determine the position of some node q using the distances to four non-coplanar points, p_0, p_1, p_2, p_3, with known position information. Let d_i denote the distance $\|q - p_i\|$, with respect to each p_i, q must lie on a sphere of radius d_i centered at p_i. In general, the intersection of the two spheres centered at p_0 and p_1 results in a circle c. Node q must lie on c. The circle c intersects the sphere centered at P_2 at two points, in general. The node q must lie on either of these two points. However, if the p_is are not coplanar, only one of these two points will be at a distance d_3 from point p_3. The coordinates of this point gives the coordinates of node q. Hence, the position of a node is uniquely determined by the distances to four non-coplanar points with known coordinates.

To start a rigid structure, we compute the all pairwise distances between some four nodes p_1, p_2, p_3, p_4 (which we assume lie at distinct points in space). The resulting structure is a tetrahedron, which is the simplest globally rigid graph in 3D. It is globally rigid no matter where the points p_i are located, but for further building, suppose that they do not lie on a common plane. Next, we compute the distances of these four points from some other point p_5; because the position of p_5 is uniquely determined, and because the four nodes p_0, p_1, p_2, p_3 form a rigid structure, the resulting structure continues to be rigid. We continue to build the graph incrementally by collecting distances to some node p_i, which is currently not on the graph, from four

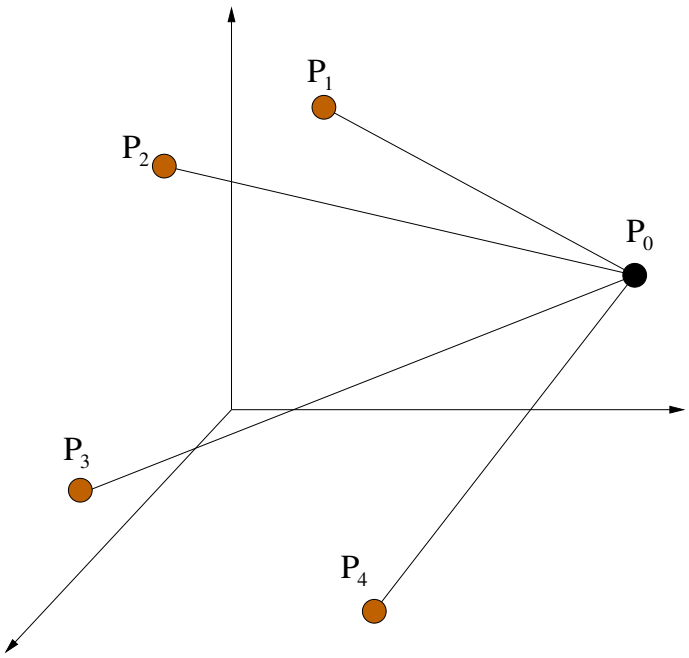


Figure 6-7: Connecting a node (p_0) to four non-coplanar points on a globally rigid graph results in a globally rigid graph.

non-coplanar nodes that are already on the graph as in Figure 6-7. This approach for building globally rigid graphs can be summarized as follows:

Theorem 6.1. *A graph is globally rigid if it is formed by starting from a clique of four non-coplanar nodes and repeatedly adding a node connected to at least four non-coplanar existing nodes.*

6.4.2 Creating a Globally Rigid 2D Structure

In 2D, we can uniquely determine the position of some node q using the distances to three non-collinear points, p_0, p_1, p_2, whose positions are known. Let d_i denote the distance $\|q - p_i\|$; with respect to each p_i, q now lies on a circle of radius d_i centered at p_i. In general, two circles intersect at two points. However, if the p_is are not collinear, only one of these two points will be at a distance d_2 from point p_2. The coordinates of this point gives the coordinates of node q. Hence, in 2D, the position of a node is uniquely determined by the distances to three non-coplanar points with known coordinates.

To build a rigid structure in 2D, we compute the all pairwise distances between some three non-collinear nodes p_1, p_2, p_3. The resulting structure is a triangle, which is globally rigid in 2D. Next, similar to the 3D graph, we incrementally build a rigid graph by computing distances to a node p_i, which is currently not on the graph, from some three non-collinear nodes that are already on the graph. This approach can be summarized as follows:

Theorem 6.2. *A 2D graph is globally rigid if it is formed by starting from a clique of three non-collinear nodes and repeatedly adding a node connected to at least three non-collinear existing nodes.*

6.4.3 Mobile Node Movement Strategy

Combining the approaches for deriving distances between stationary nodes with Theorem 6.1 describing how to build globally rigid graph, we obtain the following movement strategy for the mobile to collect distances in a 3D node deployment:

1. Initialize:
   (a) Find four stationary nodes that can all be seen from a common mobile location.
   (b) Move the mobile to at least seven nearby locations and collect distances.
   (c) Compute the pairwise distances between the four stationary nodes, using Proposition 6.3.
   (d) According to Theorem 6.1, these four nodes are on the rigid graph.
2. Loop:
   (a) Pick a stationary node n which is on the rigid graph but has not yet been examined by this loop.
   (b) Move the mobile around the visibility region of n, searching for positions from which the mobile can hear a stationary node m which is not on the rigid graph as well as zero, one, or two additional stationary nodes that are on the rigid graph.
   (c) For this collection of nodes:
      i. Compute the distances among those two or three, or four stationary nodes using Proposition 6.1, 6.2, or 6.3.
      ii. If node m now has four known distances to stationary nodes that are on the rigid graph, according to Theorem 6.1, m becomes a member of the rigid graph.

This algorithm terminates either when every stationary node is on the rigid graph (success) or when no more progress can be made according to Theorem 6.1 (failure). This algorithm makes as much progress as possible from its starting point. Furthermore, we can show that success is independent of the particular tetrahedron from which we start.

Because the mobile node stop searching for distances to a stationary node m once it has four known distances to m from nodes on the rigid graph, the number of distance measurements collected by the mobile node is linear in the number of stationary nodes.

6.5 Simulation Results

This section presents the results of running several simulations of the MAT algorithm. Although MAT has theoretical correctness and performance guarantees, it is

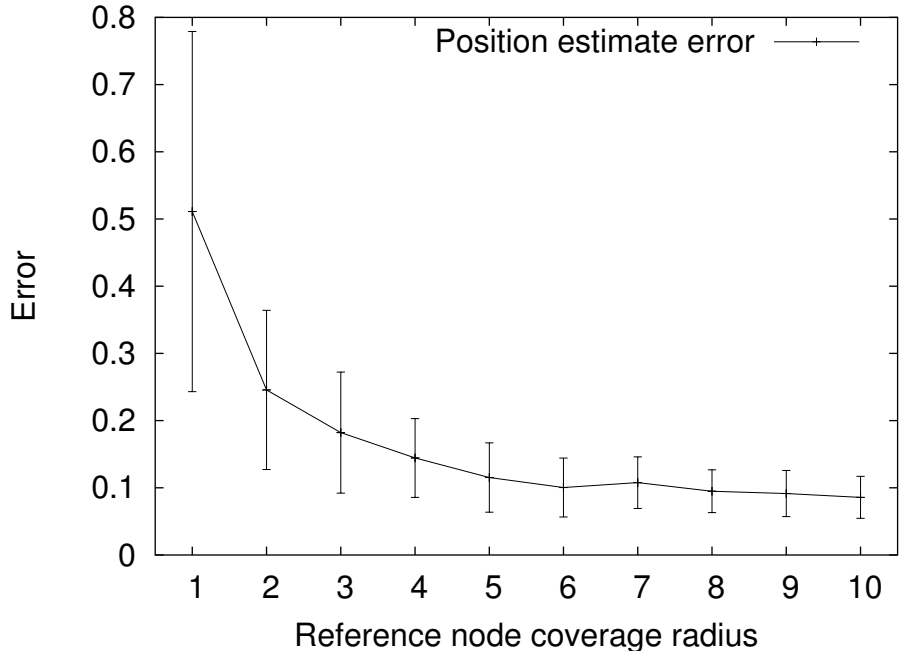


Figure 6-8: The position estimate error as a function of the radius of the reference node coverage area.

important to understand how well it performs under errors, scale, and various layout geometries.

6.5.1 Impact of GDOP on Localization Error

We start with some experiments to evaluate the impact of GDOP on location estimation using the following configuration. We have n fixed reference nodes, uniformly spaced, on a circle with radius r. We place a node m, 10 units away from the circle, on the perpendicular passing through the center of the circle. We introduce a uniformly distributed random error in the range $(-0.1, 0.1)$ units on the distances between the reference node and m. We compute the position estimate of m that minimizes the sum-squared-error for different values of r. Figure 6-8 plots the position estimate error of m, computed by the distance between the estimated and true positions, as a function of r for $n = 4$; each point on the graph represents 100 simulations. We observe that the error decreases with increasing r. Since we have kept the measurement error distribution constant, this graph shows the impact of geometry on the position estimate accuracy. It also shows the importance of reference points that cover a large area for accurate position estimation.

Figure 6-9 shows how position estimate error changes with n for $r = 10$. The position estimate error decreases with increasing n as positive and negative errors tend to cancel out with large n. This implies that we can improve position estimation accuracy using a large number of measurements.

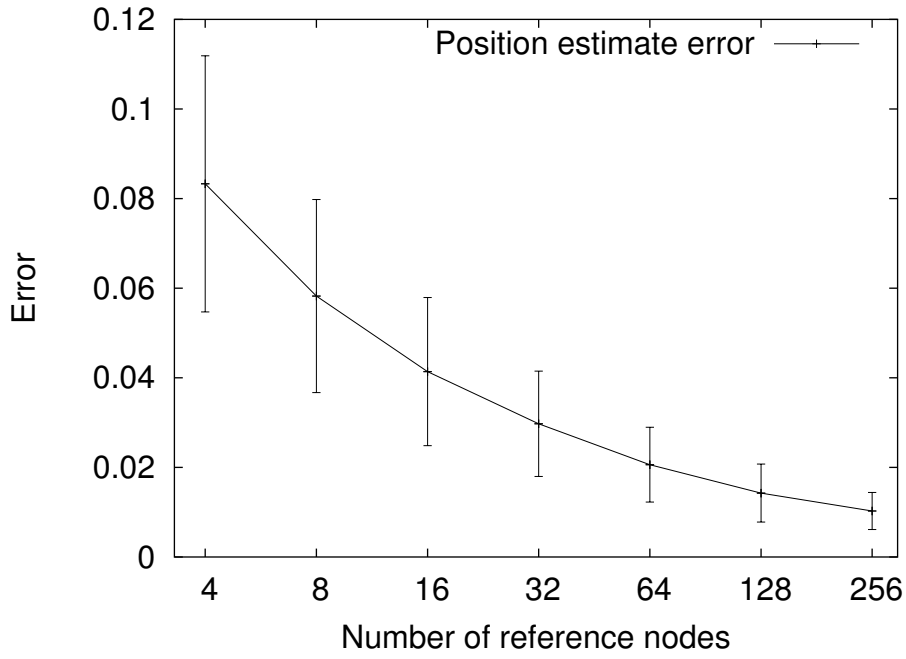


Figure 6-9: The position estimate error as a function of the number of reference nodes.

6.5.2 Mobile-Assisted Distance Estimation Performance

Next, we evaluate the performance of MAT as we vary the area covered by the mobile unit and the number of measurements. We selected three nodes, representing the fixed nodes, with (x, y) coordinates at randomly selected points on a circle of radius 10 units, with the restriction that the angle incident on the center of the circle by any two points is $> 10^o$. The z coordinates of the points were uniformly distributed between 2.5 and 5.0 units. We selected n mobile node positions uniformly distributed within a concentric circle of radius r, on $z = 0$. To achieve a uniform distribution, we placed a bounding circle of radius r' at each mobile point and iterated over different values of r'.

We examine MAT performance as we vary the mobile node coverage area. Figure 6-10 shows the average error in computing inter-node distances among three nodes as we vary r for both $n = 6$ and $n = 24$. We introduced a (1%,-1%) uniformly distributed error on mobile-to-fixed node distance estimates. Each point represents 20 simulations. We observe that a larger mobile coverage area reduces the distance estimate error. This result indicates that MAT performs better when the mobile collects samples within a large coverage area.

Next, we examine the MAT performance as we vary the number of mobile node positions. Figure 6-11 plots the average distance computation error vs n for both $\pm 1\%$ and $\pm 10\%$ uniformly distributed mobile-to-fixed node distance error. As expected, the average error decreases with increasing n. This result demonstrates a significant advantage of the MAT approach, where we can obtain a large number of mobile distance estimates at little extra cost on the infrastructure (assuming we can neglect

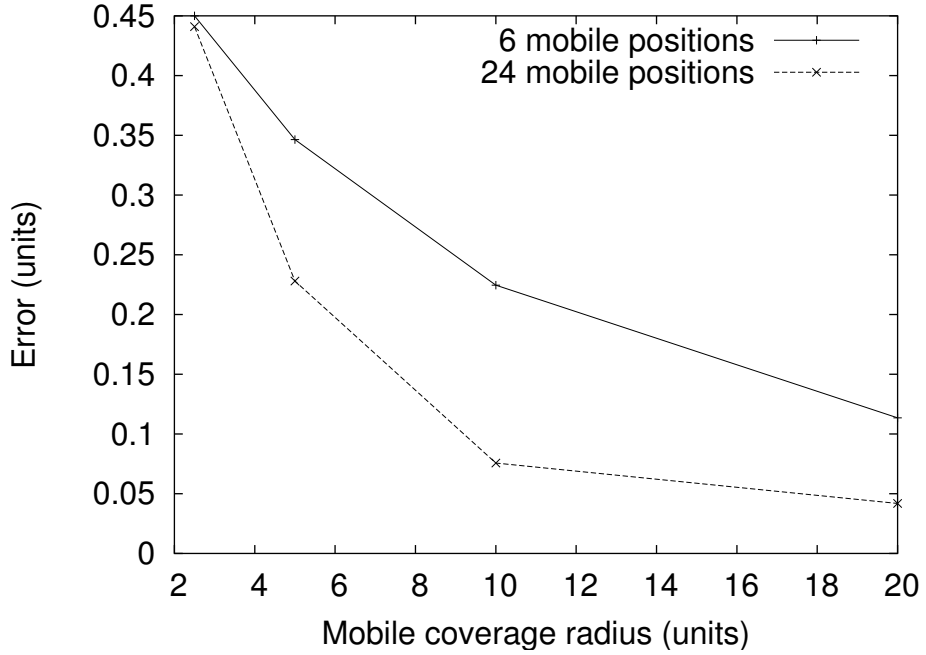


Figure 6-10: The average edge length error as a function of the radius of the mobile coverage area.

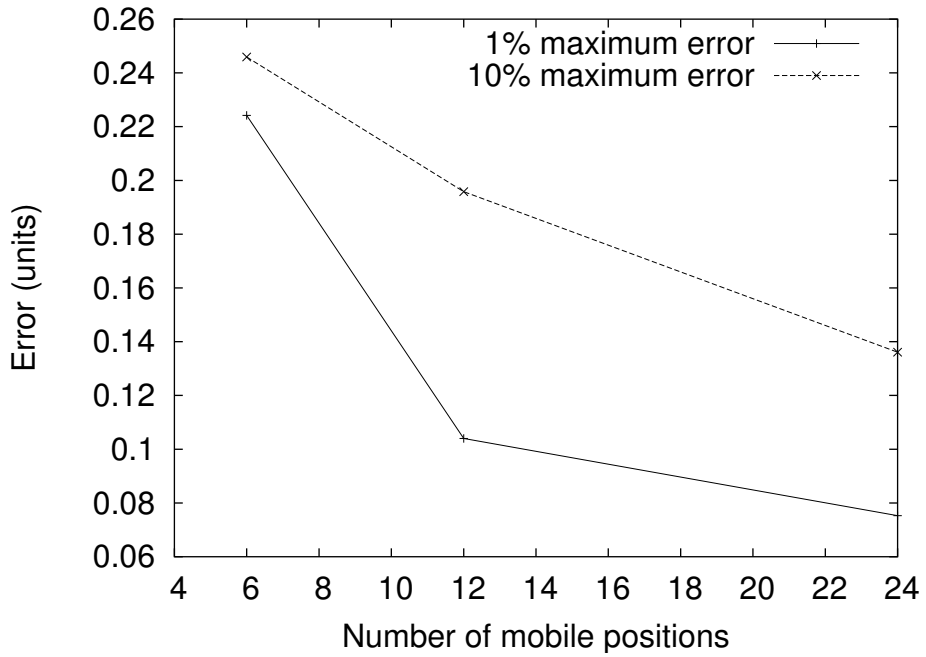


Figure 6-11: The average edge length error as a function of the number of mobile positions.

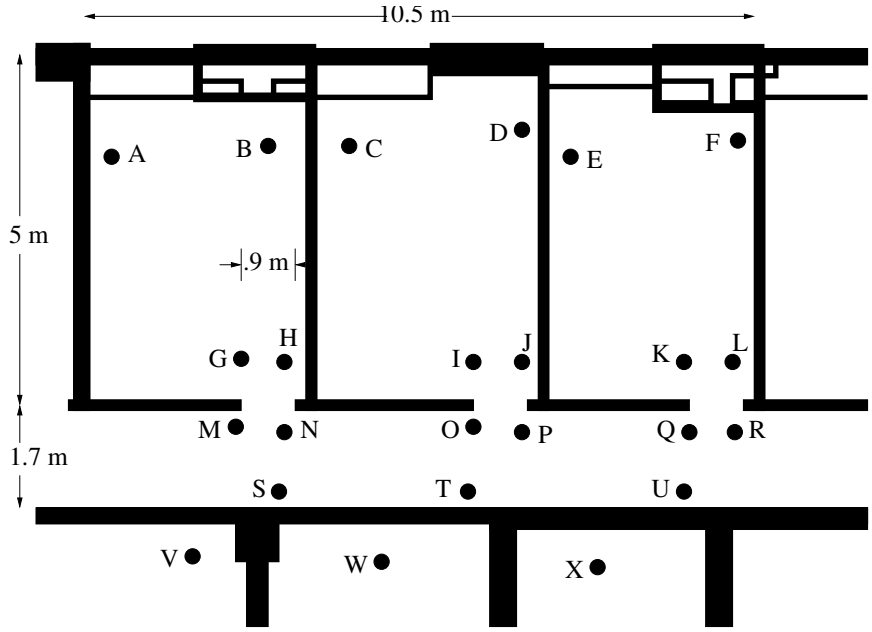


Figure 6-12: An indoor deployment of 24 nodes to evaluate the performance of MAT.

the cost associated with a mobile collecting data).

6.6 Results from a Deployment

In this section we evaluate the performance of MAT, measuring the error characteristics of the pairwise distance estimates it produces. We evaluate the performance of MAT using a 24-node beacon deployment. Figure 6-12 shows our deployment which covers four different rooms, three of these rooms are connected by a common corridor. The only line-of-sight connectivity from one room to the corridor is through the 0.9 m wide door. The rooms have no direct connectivity to each other. All the nodes except **O** and **T** were on the ceiling at the same height. Nodes **O** and **T** were attached to a beam 30 cm below the ceiling.

To compare the distances produced by MAT with the true distances between the beacons, we manually measured the distances between different walls and beacons using a laser range finder; although these distances may contain measurement errors, we will refer to the coordinates obtained from these distances as *true coordinates*, to distinguish them from the coordinates computed using MAT.

6.6.1 Inter-Beacon Distance Measurement Accuracy

We collected distance samples using a mobile listener mounted 142 cm below the ceiling. We could not collect distance samples from beacons **V**,**W**, and **X**; so we did not attempt to localize these beacons. However, these three beacons were useful for the RF connectivity based initialization phase of AFL.

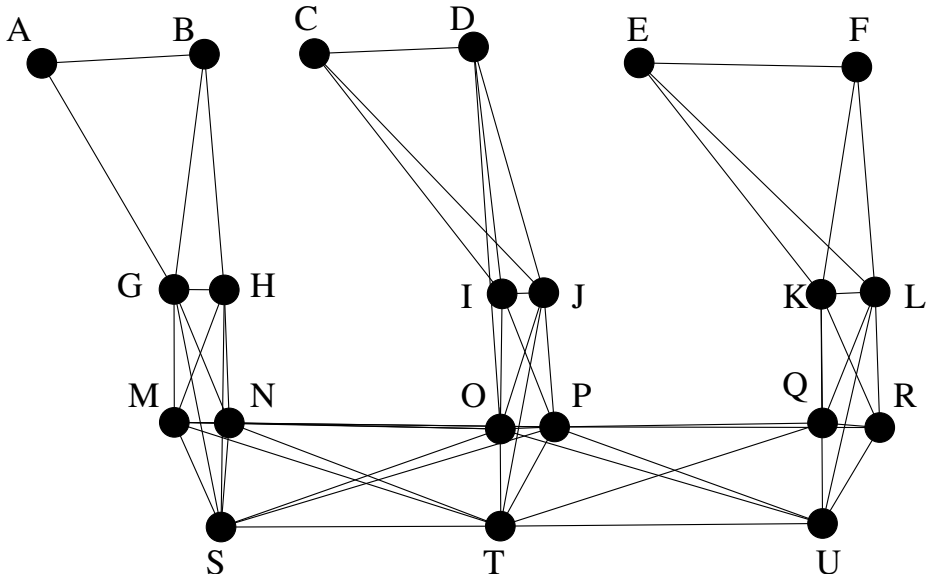


Figure 6-13: The node connectivity graph obtained by MAT. Although this graph is only locally rigid, the AFL initialization phase prevents foldings along edges such as G-H during localization.

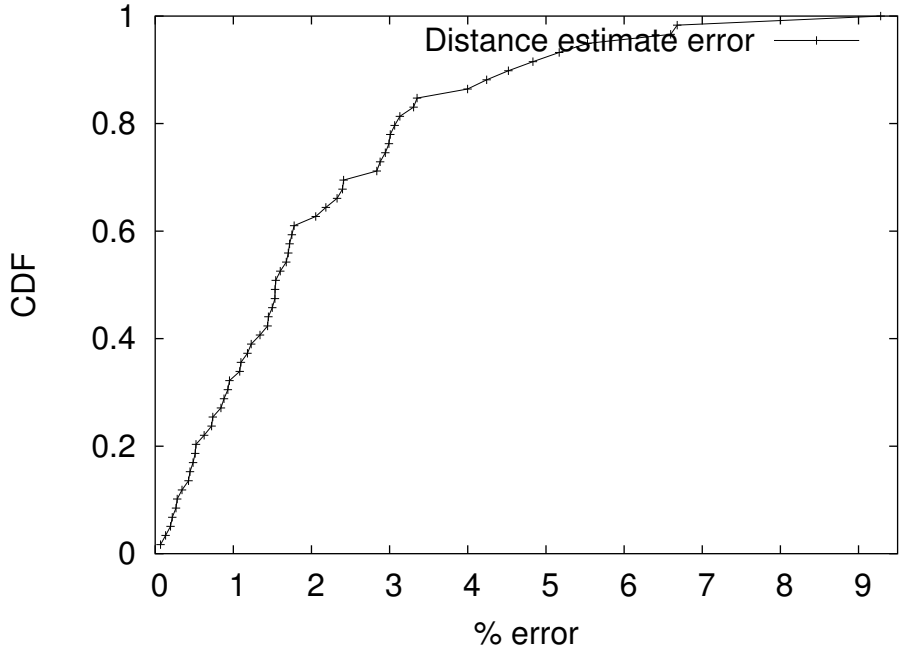


Figure 6-14: CDF of distance estimate error after the filtering and averaging for outlier rejection.

We collected distance samples by stopping the mobile listener at 1,592 points. We used the *three nodes at a time* approach to compute the inter-beacon distances. We ran the distance estimation algorithm on 52 different triangles formed by different beacon combinations. The edges of these triangles represented 59 unique edges connecting the beacons. Figure 6-13 shows the graph obtained obtained by these edges with nodes at their measured coordinates. We observe that MAT enabled us to compute enough edges to build a locally rigid graph from a collection of disconnected beacons. This graph is only locally rigid since sections of the graph can fold along edges such as K-L, B-G while preserving edge lengths. However, as we see in Chapter 7, our localization algorithm managed to avoid such folds during localization since the initialization phase of the localization algorithm generates an approximately fold-free initial coordinate assignment.

Ultrasonic propagation effects such as bending and reflection off obstacles introduces errors. We used the following solution to this problem. We had multiple distance estimates between a given pair of beacons, since an edge is typically shared by several triangles. Since the magnitude of measurement error depends on the position of the listener, we could filter out most of the outliers using a simple binning and majority election algorithm. After filtering, we computed a given edge length by averaging the estimates from different triangles.

Figure 6-14 shows the CDF of the percentage edge length error of the distance estimates obtained using MAT, after filtering and averaging to remove outliers. We observe that the distance estimation error is smaller than 1.5% over 50% of the time, and the 90th percentile has $\simeq 5\%$ error. This experiment indicates that MAT can provide accurate pairwise node information. We observe that there is a wide range of percentage error values, which we attribute to the differences in the area and the shape of individual triangles, and the restrictions on the coverage area of the listener due to physical obstacles such as furniture.

6.7 Chapter Summary

This chapter described the MAT algorithm that computes inter-beacon distances and build a globally rigid graph by collecting distances at a mobile listener. First, this chapter described the importance of global rigidity when computing node coordinate using inter-node distances. Next, this chapter examined why it is desirable to use distances collected at a mobile node to infer distances between deployed nodes. This chapter also described how to obtain inter-beacon distances using distances collected at the mobile listener for different beacon configurations. This chapter then described a mobile listener movement strategy to build a rigid graph of beacons. Finally, this chapter examined the performance of MAT using simulations and results from a real deployment. The next chapter describes the AFL algorithm which computes a beacon coordinate assignment using the inter-beacon distance estimates from MAT and the RF connectivity between beacons.

Chapter 7

Anchor-Free Localization

As described in Chapter 5, Cricket listeners use distance measurements to nearby beacons whose coordinates are known to determine their own position. Chapter 4 described how the listeners measure distances to nearby beacons accurately. Chapter 6 described how to use measurements at a mobile listener to obtain a sufficient number of inter-beacon distances, such that a beacon coordinate assignment that satisfies these distances resembles the actual layout of the beacons. This chapter describes the Anchor Free Localization (AFL) algorithm [70] for computing beacon coordinates using the inter-beacon distances obtained in Chapter 6.

A coordinate assignment that closely approximates the measured inter-beacon distances may not represent the actual beacon deployment for two reasons. First, if the collection of beacons and the inter-beacon distances do not represent a rigid graph (Section 6.1), then the resulting coordinate assignment can result in a structure that is very different from the true beacon deployment, because the set of edges do not uniquely define the relative positions of the beacons. Second, if the measurement errors are too large, a coordinate assignment that approximates the measured inter-beacon distance will not result in a structure that represents the true beacon deployment. The MAT algorithm described in Chapter 6 computes enough inter-beacon distances such that the resulting graph is rigid; our simulations and experiments showed that MAT estimates inter-beacon distance well, with a large percentage of computed edges having only a small percentage error.

In the terminology of Chapter 2 (Section 2.4) AFL is a concurrent, anchor-free localization algorithm that does not need any pre-configured node coordinates. AFL has two phases. In the first phase, AFL computes an initial coordinate assignment that results in a beacon layout similar in rough shape to the physical layout of the beacons. In the second phase, AFL iteratively updates the beacon coordinates to improve the current coordinate assignment. Our simulations show that the difference between the beacon layout obtained from AFL and the actual beacon layout—or the localization error—has the same magnitude as the inter-beacon distance measurement error. Since AFL can be used as a general localization algorithm to compute a node coordinate assignment using inter-node distances, the rest of this chapter uses the term "node" to represent a Cricket beacon or any other node whose coordinates need to be computed.

Section 7.1 introduces some terminology used in the rest of this chapter. Section 7.2 describes two theorems that guide the design of AFL. Section 7.3 describes the first phase of AFL where the shortest-path hop counts from five elected nodes are used to compute a coordinate assignment that approximates the true node layout. Section 7.4 describes the second phase of AFL, where the node coordinates are updated iteratively to improve the current coordinate assignment. Section 7.5 describes some practical aspects of using AFL with the MAT algorithm of Chapter 6. Section 7.6 uses simulations to examine various properties of the AFL algorithm, and Section 7.7 presents experimental results from a real-world deployment.

7.1 Terminology

This section describes some of the terminology used in the rest of this chapter. Section 7.1.1 identifies different types of inter-node distances, Section 7.1.2 describes the different types of error that result from differences between different types of distances, and Section 7.1.3 describes *order-correctness*—a property that describes the similarity between two graphs based on the spatial ordering of nodes.

7.1.1 Types of Distances

We identify three types of distances between the nodes in a graph of n nodes deployed in 3D space, where each node i is assigned the coordinates (x_i, y_i, z_i) within some coordinate system.

The *true distance*, $d_{\mathrm{T}}(i,j)$, between nodes (i,j) is the actual physical distance between these two nodes as defined by their positions in 3D space. The value $d_{\mathrm{T}}(i,j)$ is defined for all i,j. For two nodes (i,j), the *computed distance*, $d_{\mathrm{C}}(i,j)$, is the distance computed from the current coordinate assignment of i and j; the computed distance is also defined for all i,j. The *measured distance*, $d_{\mathrm{M}}(i,j)$ between nodes (i,j), is the distance obtained from some ranging technique. Because all pairwise distance measurements may not be available, $d_{\mathrm{M}}(i,j)$ is defined only for a subset of the nodes (i,j). Because all ranging techniques introduce some measurement error, $d_{\mathrm{M}}(i,j) \neq d_{\mathrm{T}}(i,j)$. If the measured distance $d_{\mathrm{M}}(i,j)$, between two nodes (i,j), is defined (available), then the two nodes (i,j) are called *neighbors*; this neighborhood relationship is denoted by $i \leftrightarrow j$, and by placing an edge between the nodes i and j.

In Cricket, $d_{\mathrm{M}}(i,j)$ corresponds to the inter-beacon distances obtained from MAT (Chapter 6). This chapter uses the term *measured distance* to represent the inter-beacon distances obtained from MAT, although these distances are obtained indirectly from the distances measurements at a mobile listener.

7.1.2 Representing Error in a Coordinate Assignment

For a given pair of nodes $i,j, i \leftrightarrow j$, the difference $d_{\mathrm{M}}(i,j) - d_{\mathrm{T}}(i,j)$ gives the distance *measurement error*, while the value $e(i,j) = d_{\mathrm{C}}(i,j) - d_{\mathrm{M}}(i,j)$ gives the distance

computation error. For the rest of this chapter, we use the term *error* to represent the computation error.

The *computed error vector* (or simply the *error vector*) on i due to j, denoted by $\vec{e}(i,j)$, is given by $\vec{e}(i,j) = e(i,j)\hat{u}_{ij}$, where $\hat{u}_{ij}$ is the unit vector from i to j obtained from the current coordinates of i and j. If nodes i and j have the same coordinate assignment, $\vec{e}(i,j)$ is undefined (hence, not used in calculations). Under this definition, $\vec{e}(i,j)$ *pulls* i toward j when $d_{\mathrm{C}}(i,j) > d_{\mathrm{M}}(j,i)$, and $\vec{e}(i,j)$ *pushes* i away from j when $d_{\mathrm{C}}(i,j) < d_{\mathrm{M}}(i,j)$. We observe that updating the coordinate assignment of i in the direction of $\vec{e}(i,j)$ by a distance $d < |\vec{e}(i,j)|$ reduces $|\vec{e}(i,j)|$.

The resultant error vector on node i, $\vec{e}(i)$ is given by,

$$\vec{e}(i) = \sum_{j:i\leftrightarrow j} \vec{e}(i,j). \tag{7.1}$$

The *sum squared error* of node i, $E_{ss}(i)$, is given by,

$$E_{\mathrm{ss}}(i) = \sum_{j:i\leftrightarrow j} |e(i,j)|^2. \tag{7.2}$$

Consider a graph G that consists of n nodes and the edges $(i,j), \forall i \leftrightarrow j$ The sum squared error of G, $E_{\mathrm{ss}}(G)$, is given by,

$$E_{\mathrm{ss}}(G) = \frac{1}{2}\sum_{i\leftrightarrow j} |e(i,j)|^2. \tag{7.3}$$

For a graph G, where $d_{\mathrm{M}}(i,j) = d_{\mathrm{T}}(i,j), \forall i \leftrightarrow j$, there is some coordinate assignment where $d_{\mathrm{C}}(i,j) = d_{\mathrm{M}}(i,j) = d_{\mathrm{T}}(i,j)$, which results in $E_{\mathrm{ss}}(G) = 0$. Because that coordinate assignment satisfies the true distances, assuming G is globally rigid, that coordinate assignment results in a graph that is identical to the true embedding of the nodes modulo rotation and translation.

Usually it is not possible to obtain a coordinate assignment that satisfies $E_{\mathrm{ss}}(G) = 0$, since $d_{\mathrm{M}}(i,j) \neq d_{\mathrm{C}}(i,j)$ in general (although the given node layout satisfies the true distances, there may not be a node layout that satisfies the measured distances). When measurement errors are small, the coordinate assignment corresponding to the global minimum of $E_{\mathrm{ss}}(G)$ usually results in a graph that approximates the true node layout well. In the rest of this chapter we assume $d_{\mathrm{M}}(i,j) = d_{\mathrm{T}}(i,j)$ so there is some coordinate assignment that satisfies the measured distances, and the global minimum of $E_{\mathrm{ss}}(G) = 0$. Later, in Section 7.6, we revisit the topic of non-zero measurement error to evaluate the performance of AFL in the presence of measurement errors.

7.1.3 Order-Correct Graphs

The two graphs G and G' with the same set of nodes and edges, but with possibly different edge lengths, are called *order-correct* with respect to each other if there is *some* Cartesian coordinate system $X - Y - Z$ in G, and $X' - Y' - Z'$ in G' that satisfies the following conditions:

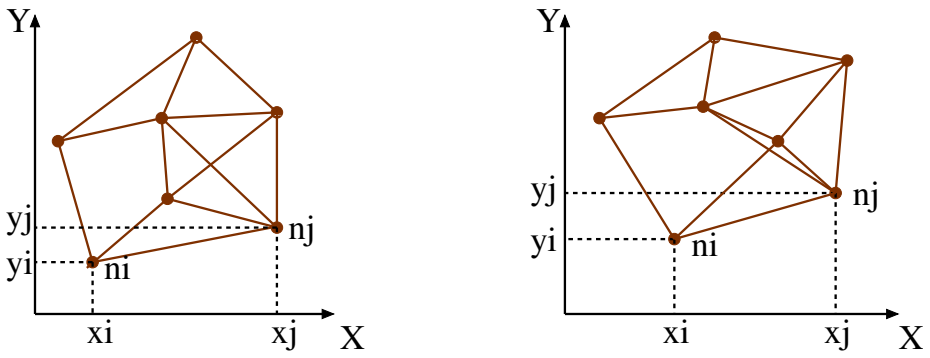


Figure 7-1: Two graphs with identical nodes and edges, but different edge lengths, that are order-correct with respect to each other.

$$
\begin{aligned}
&\forall i, j : i \leftrightarrow j \\
&\quad \text{if } x_i < x_j \quad \text{then} \quad {x'}_i < {x'}_j \\
&\quad \text{if } x_i > x_j \quad \text{then} \quad {x'}_i > {x'}_j \\
&\quad \text{if } y_i < y_j \quad \text{then} \quad {y'}_i < {y'}_j \\
&\quad \text{if } y_i > y_j \quad \text{then} \quad {y'}_i > {y'}_j \\
&\quad \text{if } z_i < z_j \quad \text{then} \quad {z'}_i < {z'}_j \\
&\quad \text{if } z_i > z_j \quad \text{then} \quad {z'}_i > {z'}_j
\end{aligned}
$$

Order-correctness implies that the nodes in G and G' have an equivalent partial ordering along the axes of the two coordinate systems. For example, Figure 7-1 shows two 2D graphs that are order correct with respect to each other in 2D Cartesian coordinates. Two order-correct graphs *look similar* since their nodes have an equivalent spatial ordering.

We use the *degree of order-correctness* as a measure of the fraction of edges that are correctly ordered between two graphs, within some given coordinate axes. For example, two graphs G and G' are 100% order-correct if they are order-correct with respect to each other. However, if only 90% of the edges in G and G' are correctly ordered with respect to some given coordinate axes, then we say that G and G' are only 90% order-correct with respect to those axes.

7.2 Theoretical Framework

The AFL algorithm is guided by the following two theorems on graphs whose node positions are determined by the current coordinate assignments. We prove these two theorems using 3D Cartesian coordinate systems. However, these theorems generalize to other 3D and 2D coordinate systems as well (e.g. polar coordinates).

Theorem 7.1. *For a node i with $\vec{e}(i) \neq 0$ in some graph G, we can update the*

position of i in a direction $\vec{u}$ at an angle $\theta < \pi/2$ with respect to the direction of $\vec{e}(i)$, such that $E_{ss}(G)$ decreases after the update.

Proof. For the given graph G, without loss of generality, we select a coordinate system such that the x axis of the coordinate system lies in the direction of $\vec{u}$. Consider node i with coordinates (x, y, z). This node has m neighbors, with each neighbor j at position (x_j, y_j, z_j). The line connecting (x, y, z) and (x_j, y_j, z_j) is at an angle α_j to the x axis. The sum-squared error of node i, $E_{ss}(i)$, is given by,

$$E_{ss}(i) = \frac{1}{2} \sum_{j:i \leftrightarrow j} \left(\sqrt{(x - x_j)^2 + (y - y_j)^2 + (z - z_j)^2} - d_T(i, j) \right)^2$$

The partial derivative of $E_{ss}(i)$ with respect to x is given by,

$$\begin{aligned} \frac{\partial E_{ss}(i)}{\partial x} &= \sum_{j:i \leftrightarrow j} - \left(\sqrt{(x - x_j)^2 + (y - y_j)^2 + (z - z_j)^2} - d_T(i, j) \right) . \cos(\alpha_j) \\ &= \sum_{j:i \leftrightarrow j} - \left(d_C(i, j) - d_T(i, j) \right) . \cos(\alpha_j) = -e_x(i). \end{aligned} \tag{7.4}$$

Since the component of error vector, $\vec{e}(i)$, in the direction of x, $e_x(i)$, is > 0,

$$\frac{\partial E_{ss}(i)}{\partial x} < 0. \tag{7.5}$$

Therefore, $E_{SS}(i)$ decreases in the direction of x. Since $E_{SS}(G) = \sum_i E_{SS}(i)$, $E_{SS}(G)$ decreases when the position of node i is updated along x. □

This theorem gives the possible directions in which the coordinates of a node i can be updated to reduce $E_{ss}(G)$ of the graph G. However, this theorem does not give the step size (the distance) by which the coordinates should be updated. A linear search along the specified direction can be used to determine the position update step size that reduces $E_{ss}(i)$ (and, hence, $E_{ss}(G)$).

Lemma 7.1. *Consider two nodes p and q, $p \leftrightarrow q$, with $z_p \neq z_q$, on some 3D graph G. Assume that the current coordinate assignment results in $d_C(p, q) = d_T(p, q)$, hence $|\vec{e}(p, q)| = 0$. Suppose we update the coordinates of q along the z axis, such that the resulting graph G' is order-correct with respect to G. Then, the z component of $\vec{e}(p, q)$, in G', opposes the direction of q's coordinates update.*

Proof. There are only two possibilities for updating the coordinates of q in the z direction while preserving order correctness between G and G'. The coordinates can either be updated in the direction of -ve z, as shown in Figure 7-2 such that $d_C(p, q)$ decreases, or the the coordinates can be updated in the direction of +ve z as shown in Figure 7-2 such that $d_C(p, q)$ increases. From Figures 7-2 7-3, we observe that the z component of $\vec{e}(p, q)$, at the new positions of q, always opposes the direction of the coordinates update. □

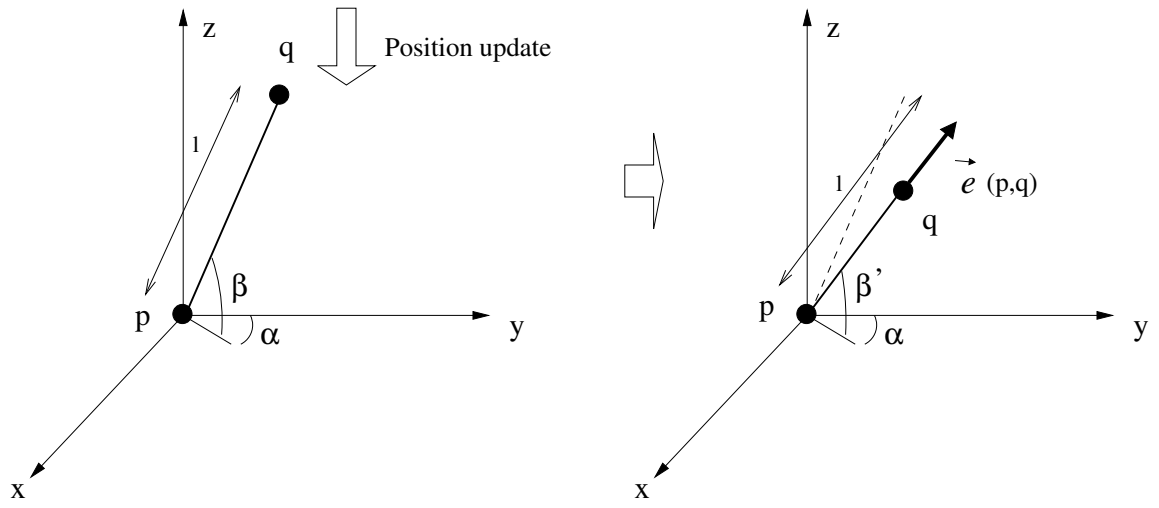


Figure 7-2: When node q's coordinates are updated in -ve z, $\vec{e}(p,q)$ has a +ve z component.

Lemma 7.2. *Consider two nodes p and q, $p \leftrightarrow q$, of some 3D graph G. Assume that $d_{\mathrm{C}}(i,j) = d_{\mathrm{T}}(i,j)\ \forall i,j : i \leftrightarrow j$, such that $\vec{e}(i,j) = 0\ \forall i,j : i \leftrightarrow j$. Suppose we update the coordinates of q in the direction of the z axis, and obtain the graph G', such that G' is order-correct with respect to G. If node coordinates updates are allowed only in the direction of the z axis, any subsequent node coordinates update in G' to balance the z component of the error vectors $\vec{e}(p)$ and $\vec{e}(q)$, while keeping the coordinates of p and q fixed and maintaining the order-correctness of the resulting graph, results in a graph G'' with at least two nodes r and s such that $\vec{e}(r) \neq 0$ and $\vec{e}(s) \neq 0$.*

Proof. Assume $z_q > z_p$ in G, and assume q's coordinates are updated in the +ve z direction. According to lemma 7.1, the error vector $\vec{e}(q)$ in G' has a -ve z component. Next we examine how to balance the z components of $\vec{e}(q)$. Error vector $\vec{e}(q)$ can be balanced by updating the coordinates of a neighbor i of q with $z_i > z_q$ in +ve z direction by a sufficient amount, such that $d_{\mathrm{C}}(i,q) > d_{\mathrm{T}}(i,q)$. Vector $\vec{e}(q)$ can also be balanced by updating the coordinates of a neighbor j of q with $z_j < z_q$ in +ve z direction by a sufficient amount, such that $d_{\mathrm{C}}(i,q) < d_{\mathrm{T}}(i,q)$ (or by a combination of coordinates updates of a multiple of such neighbors). However, any such coordinates update of a node i will result in a $\vec{e}(i)$ with a -ve z component. The resulting error vector $\vec{e}(i)$, can be balanced by updating the coordinates of a neighbor of i in the +ve z direction, which again results in a error vector with a -ve z component. Since G has only a finite number of nodes, after a sequence of such node updates in the +ve z direction, there will be at least one node having a non-zero resultant error vector with a -ve z component.

When the coordinates of q are updated in the +ve z direction, the error vector $\vec{e}(p,q)$ on p due to q has a +ve z component. Hence the error vector $\vec{e}(p)$ has a +ve z component. Similar to $\vec{e}(q)$, balancing the z component of $\vec{e}(p)$ requires a sequence of node coordinates updates in the -ve z direction. Since there are only a finite number of nodes, there will be at least one node having a resultant error with a

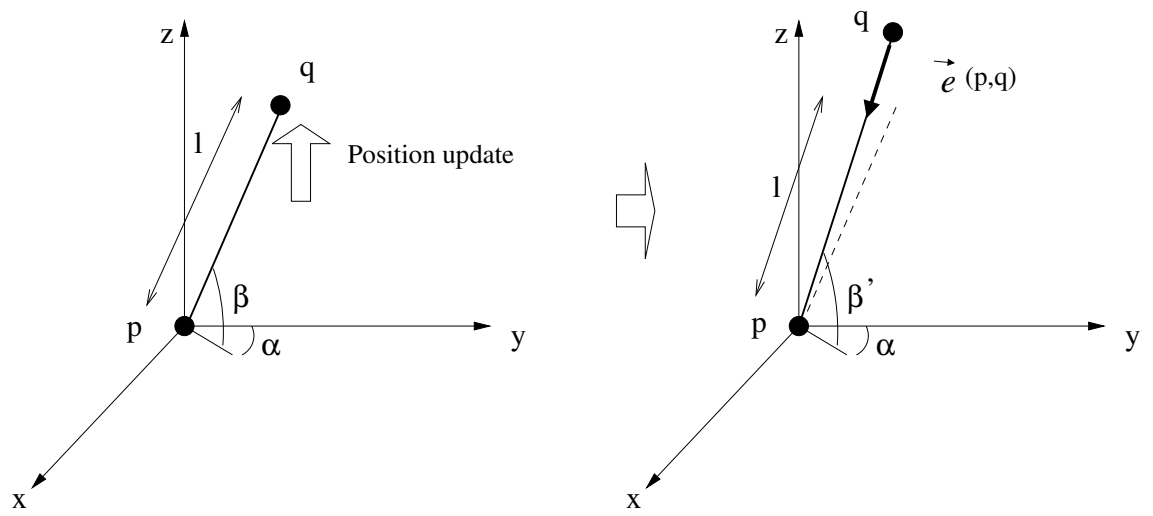


Figure 7-3: When node q's coordinates are updated in +ve z, $\vec{e}(p,q)$ has a -ve z component.

+ve z component. The same argument applies when q is moved in the -ve z direction as well. □

Next, the proposition 7.1 argues that lemma 7.2 continues to be valid when node coordinates updates are allowed in the directions of all three coordinate axes x, y, and z.

Proposition 7.1. *Consider two nodes p and q, $p \leftrightarrow q$, of some 3D graph G. Assume that $d_{\mathrm{C}}(i,j) = d_{\mathrm{T}}(i,j)\ \forall i,j : i \leftrightarrow j$, such that $\vec{e}(i,j) = 0\ \forall i,j : i \leftrightarrow j$. Suppose we update the coordinates of q in the direction of the z axis, and obtain the graph G', such that G' is order-correct with respect to G. If node coordinates updates are allowed in all the directions x, y, and z, any subsequent node coordinates update in G', while maintaining the order-correctness of the resulting graph, aimed at balancing $\vec{e}(q)$ and $\vec{e}(p)$ results in a graph G'' with at least one node r such that $\vec{e}(r) \neq 0$. All the graphs G, G', and G'' are assumed to be globally rigid.*

Proof. We first consider 2D graphs with coordinate axes z and x. When coordinates of q are updated in +ve z direction, similar to the node coordinates update in lemma 7.2, $\vec{e}(q)$ will have a -ve z component and $\vec{e}(p)$ will have a +ve z component. Next, we examine possible approaches for balancing $\vec{e}(q)$ and $\vec{e}(p)$.

We observe that, similar to lemma 7.2, the z components of the resultant errors on q and p can be balanced by a sequence of node coordinates updates in +ve z and -ve z directions. The x components of $\vec{e}(p)$ and $\vec{e}(q)$ can be balanced by coordinates updates in x direction. After the node coordinates update, similar to lemma 7.2, there will be at least two nodes r and s with non-zero error components in the z direction.

Unlike in lemma 7.2, it is possible to balance $\vec{e}(q)$ by two nodes r and s with zero (or even -ve) coordinates updates in +ve z direction, but whose coordinates are updated in -ve x and +ve x directions such that $d_{\mathrm{C}}(r,q) > d_{\mathrm{T}}(r,q)$ and $d_{\mathrm{C}}(s,q) >$

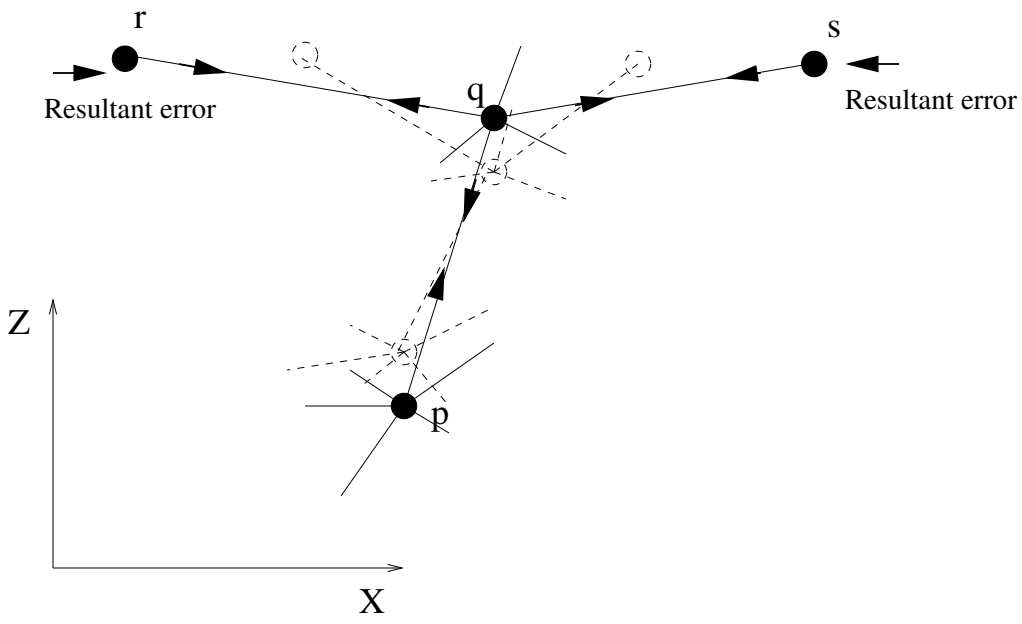


Figure 7-4: When the coordinates of q are updated in +ve z direction, the resulting $\vec{e}(q)$ can be balanced by two nodes, r and s, whose coordinates are updated in -ve x and +ve x directions, such that $d_{\mathrm{C}}(r,q) > d_{\mathrm{T}}(r,q)$ and $d_{\mathrm{C}}(s,q) > d_{\mathrm{T}}(s,q)$. However, $\vec{e}(r)$ and $\vec{e}(s)$ will have non-zero components in +ve x and -ve x directions respectively.

$d_{\mathrm{T}}(s,q)$ (Figure 7-4). Similarly, it is possible to balance the $\vec{e}(q)$ using two nodes u and v whose coordinates are updated in +ve and -ve x direction such that $d_{\mathrm{C}}(u,q) < d_{\mathrm{T}}(u,q)$ and $d_{\mathrm{C}}(v,q) < d_{\mathrm{T}}(v,q)$ (Figure 7-5). However, in both these cases, the pair of nodes whose coordinates are updated in the x direction will have resultant error vectors with non-zero x components. Hence, although this node coordinates update scheme can balance $\vec{e}(q)$, it results in two nodes having error vectors with non-zero x components.

Unlike in lemma 7.2, it is also possible for $\vec{e}(p)$ and $\vec{e}(q)$ to be completely balanced by a "string" of nodes $a, b, c, \ldots$ whose coordinates are updated in the +ve x direction, and another string of nodes $r, s, t, \ldots$ whose coordinates are updated in the -ve x direction as shown in Figure 7-6. Although this node coordinates update may completely balance $\vec{e}(p)$ and $\vec{e}(q)$, at least one node on the string $a, b, c, \ldots$ will have a resultant error vector with a -ve x component. Similarly, at least one node on the string $r, s, t, \ldots$ will have a resultant error vector with a +ve x component. Thus, although $\vec{e}(p)$ and $\vec{e}(q)$ are completely balanced, there are at least two nodes with non-zero error vectors.

A special situation occurs when p and q have the same x coordinates as in Figure 7-7, here a single node r, whose coordinates are updated in x direction may balance the error vectors on both p and q. However, this will result in $\vec{e}(r)$ having a non-zero component in the x direction, unless r has no other neighbors than p and q (or other neighbors that have the same x coordinates as p and q). However, a node r with only two neighbors (or more than two neighbors that lie on a line) makes the graph non-rigid or locally rigid, since r can assume two possible positions with respect to

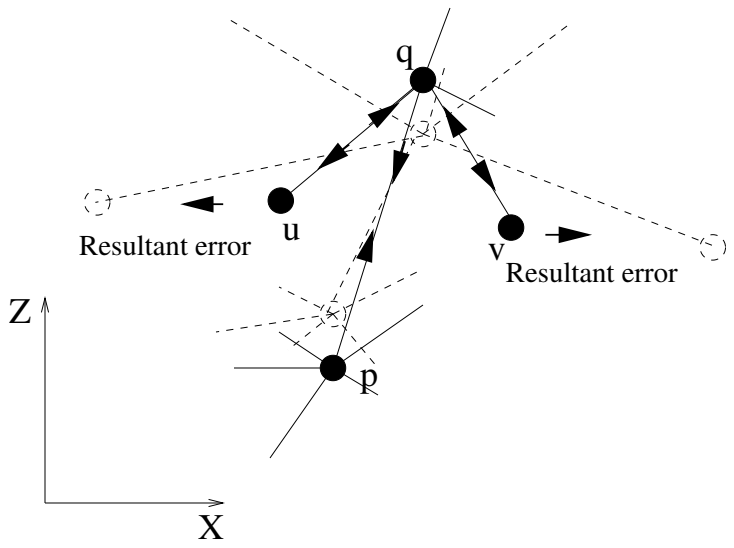


Figure 7-5: When the coordinates of q are updated in +ve z direction, the resulting $\vec{e}(q)$ can be balanced by two nodes, u and v, whose coordinates are updated in +ve x and -ve x directions, such that $d_{\mathrm{C}}(u,q) < d_{\mathrm{T}}(u,q)$ and $d_{\mathrm{C}}(v,q) < d_{\mathrm{T}}(v,q)$. However, $\vec{e}(u)$ and $\vec{e}(v)$ will have non-zero components in -ve x and +ve x directions respectively.

its neighbors for the given edge lengths. This contradicts our initial assumption of a globally rigid graph.

Thus, we can use one or more of the above techniques to balance $\vec{e}(q)$ and $\vec{e}(p)$ due to the coordinates update of q in the +ve z direction. However, this will result in at least one node with a non-zero resultant error. A sequence of coordinates updates to balance the resulting error will still result in at least one node having a non-zero resulting error.

Above proof assumes a 2D graph. The proof can be extended to 3D graphs as follows. The resultant error on q and p can be balanced either by nodes whose coordinates are updated in +ve and -ve z direction, or by one or more nodes whose coordinates are updated in x and y directions. However, after a sequence of node coordinates updates, there will be at least one node with a non-zero resultant error. □

We can come up with the following hypothetical algorithm to solve the localization problem using theorem 7.1 and proposition 7.1. For a graph G of n nodes, assume that we can compute an order-correct initial coordinate assignment. Once such a coordinate assignment is available, proposition 7.1 tells us that there is some node i with a non-zero resultant error vector. Now we can apply theorem 7.1 to node i to reduce the sum-squared-error of G. Hypothetically, the repeated application of proposition 7.1 will result in reaching the global minimum of $E_{ss}(G)$. However, this hypothetical algorithm is flawed due to following reasons. First, there is no known technique to obtain an order-correct initial coordinate assignment of an embedded graph using only the inter-node distances. Second, even if we start with an order-correct coordinate assignment, we cannot repeatedly apply these two theorems

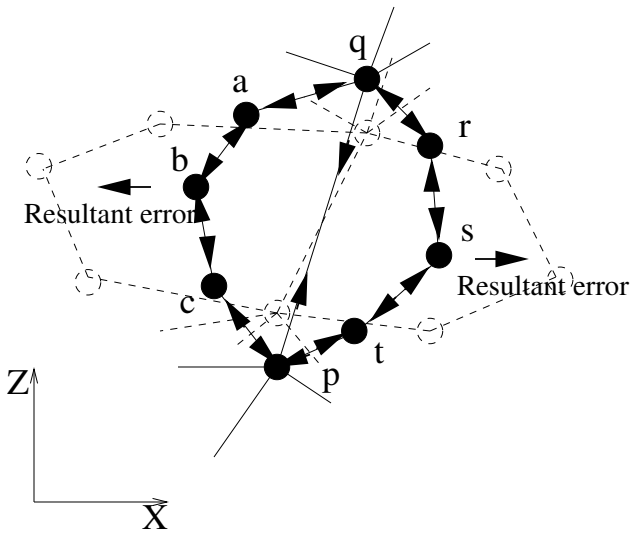


Figure 7-6: The error vectors $\vec{e}(q)$ and $\vec{e}(p)$ due to coordinates update of q in +ve z direction can be balanced by the two strings of nodes (a, b, c) and (r, s, t) with coordinates updates in +ve x and -ve x directions respectively. However, there will be at least one node among (a, b, c) having an error vector with a non-zero component in the -ve x direction, and at least one node among (r, s, t) having an error vector with a non-zero component in the +ve x direction.

until we reach the global minimum of $E_{ss}(G)$, because the repeated application of theorem 7.1 may result in a graph whose node positions can no longer be updated according to theorem 7.1, while preserving the order correctness of G.

AFL mimics the behavior of the above hypothetical algorithm as follows. AFL first computes an initial coordinate assignment that results in a scaled-up "approximately order-correct" version of the true node layout. This initial coordinate assignment is approximately order-correct because the fraction of edges that violate order-correctness is small. After the initial coordinate assignment, AFL iteratively applies a modified version of theorem 7.1—with a limited step size and no testing for order violations—to minimize the sum-squared error $E_{ss}(G)$. As the Section 7.4 describes, the limited step size, combined with the scaled-up initial coordinate assignment, reinforces the order-correctness of the graph, resulting in an increased likelihood of reaching the global minimum of $E_{ss}(G)$.

7.3 AFL Phase-1: Hop Count-Based initialization

The AFL phase-1—the initialization phase—computes a coordinate assignment resulting in an approximately order-correct graph in a *polar coordinate* system with respect to the physical layout of the n nodes. The coordinate assignment is only approximately order-correct because there is a small percentage of edges that may violate the ordering. Saxe has shown that coming up with a correct ordering based on inter node distances alone is NP-hard even in 1D space [86]. We use the following

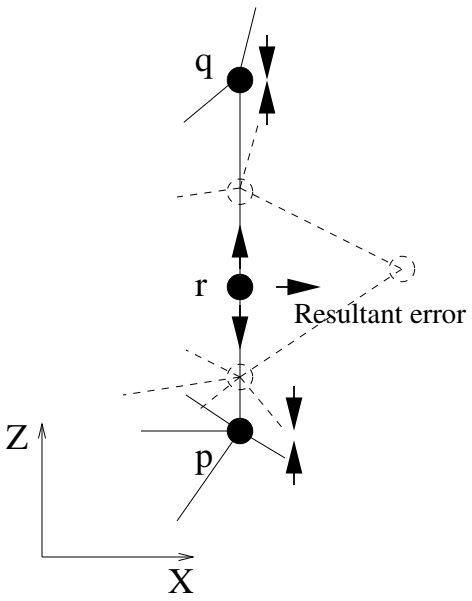


Figure 7-7: The error vectors $\vec{e}(p)$ and $\vec{e}(q)$ due to coordinates update of q may be balanced by the coordinates update of a single node r. This results in a $\vec{e}(r)$ with a non zero component in the direction of +ve x, unless all the neighbors of r (including p and q) have the same x coordinates. However, this condition causes the original graph to be only locally rigid.

assumption to simplify the computation of a *mostly-order-correct* initial coordinate assignment.

Assumption 1. *There is some value R, called the "range" of the graph ,such that $\forall i, j, i \leftrightarrow j$ if and only if $d_{\mathrm{T}}(i, j) \leq R$.*

As we describe below, with this assumption, the shortest-path hop count from a given node i approximates the ordering of the rest of nodes with respect to i well. However, in a practical node deployment, the above assumption may not always hold because obstacles between nearby nodes may prevent distance between nodes that are less than R apart from being measured. We later observe that, even when we represent the range as a random variable uniformly distributed within a given range, the shortest-path hop count continues to reflect the ordering of nodes with respect to a given node well.

AFL phase-1 uses shortest-path hop counts from multiple nodes to obtain an initial coordinate assignment that approximates the ordering of nodes in the physical layout.

7.3.1 Hop Count as a Measure of Euclidean Distance

During AFL phase-1, we use the shortest path hop counts between nodes to come up with an initial coordinate assignment that is approximately order-correct with respect to the true node layout. In this section we examine the performance of the

shortest-path hop count as a measure of how nodes are ordered with respect to a given node. In particular, given some node i, we examine how well the shortest-path hop counts, $h_{p,i}$ and $h_{q,i}$ to two nodes p and q from i, represent the location of p and q with respect to node i. Ideally, we require $h_{p,i} > h_{q,i} \iff d_{\mathrm{T}}(p,i) > d_{\mathrm{T}}(q,i)$ and vice versa.

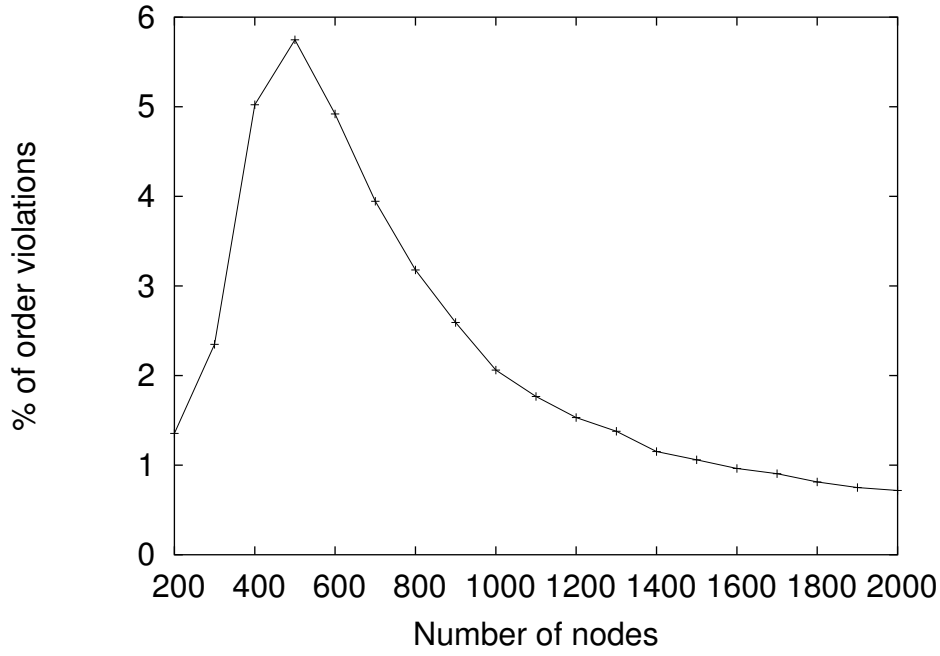


Figure 7-8: Fraction of order violations, with respect to a node located at the center of a circle with a radius of 10 (units), as a function of the number of nodes deployed within the circle. Nodes have a fixed range of 1 (unit).

We used the following simulation to examine how well the shortest-path hop counts reflect the location of two nodes with respect some given node. We selected a circle C of radius 10 units, with the center of C representing the origin of the Cartesian coordinate system. We uniformly deployed n nodes inside C as follows. The x and y coordinates of each node were selected as random variables with a uniform distribution over the range $(-10, 10)$; only the (x, y) combination that fell within C caused a node to be added. We added the node, n_0, at the center of C. Next, using a range equal to 1, we added an edge between each pair of nodes separated by a distance less than 1. Next we computed the shortest-path spanning tree rooted at n_0 and obtained the shortest path hop count $h_{i,0}$ to each node i from n_0. Then we counted the number of edges (p, q) such that $h_{p,0} < h_{q,0}$ and $d_{\mathrm{T}}(p, 0) > d_{\mathrm{T}}(q, 0)$—these are the pairs of nodes whose ordering is violated with respect to n_0, when shortest-path hop count from n_0 is used as a measure of the true distance. We computed the percentage of such violations with respect to all the edges on the tree rooted at n_0. For each value of n, we ran 100 simulations. Figure 7-8 plots the average value of the fraction of edges that violate the ordering with respect to n_0, as a function of the number of nodes n.

We observe that, overall, the percentage of order violations is small. The percentage of order violations increase with n until n reaches 500, and then the percentage of violations start to drop. We believe that the small number of initial order violations is due to the small number of nodes that lie on the graph rooted at n_0 for small n due to low connectivity. As n increases, the number of nodes that lie on the tree increases, resulting in the initial increase in order violations with n. When $n > 500$, almost all the nodes lie on the tree rooted at n_0, and the number of order violations decreases with increasing n due to the increased connectivity. However, we observe that the percentage of order violations is less than 6 %. In particular, for large n, these simulations indicate that the shortest-path hop-count can be used as a good measure to determine the spatial ordering of neighboring node pairs with respect to a given node.

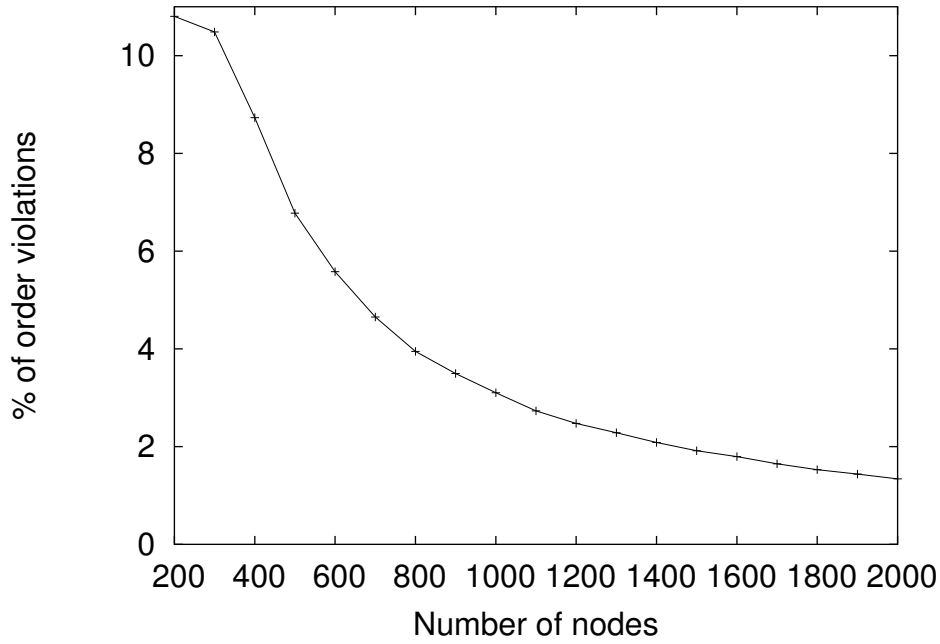


Figure 7-9: Fraction of order violations, with respect to a node located at the center of a circle with a radius of 10 (units), as a function of the number of nodes deployed within the circle. Nodes have a variable range that is uniformly distributed over (0,2) (units).

Next, we used simulations to examine the effect of having a variable value for the range. We selected the range as a uniformly distributed random variable r, distributed within the range $(0, 2)$ units (giving an average value of 1). For a given pair of nodes p, q, we obtained an instance of the random variable r, and added the edge (p, q) if $d_{\mathrm{T}}(p, q) < r$. We repeated the experiment described above and calculated the fraction of edges that violate the ordering. Figure 7-9 plots the average value of the fraction of violate with respect to n_0 as a function of n. Here we observe that, in contrast to the fixed range, the random range results in a large number of violations for small n. However, for large n, the random range has almost identical performance as the

fixed range. We explain this behavior as follows. Using a random value for the range results in a smaller number of edges compared to the fixed-range simulations. When n is small, the number of edges in the graph is small, and when a subset of these edges are removed randomly, it is likely that some of the nodes that are closer to n_0 are being connected to n_0 through nodes that are further away from n_0, resulting in an increased number of order violations. However, when n is large, the graph has sufficient connectivity, such that the removal of a subset of the edges does not significantly affect the availability of shortest paths that radiate out from n_0.

7.3.2 AFL Phase-1: Initial Coordinate Assignment

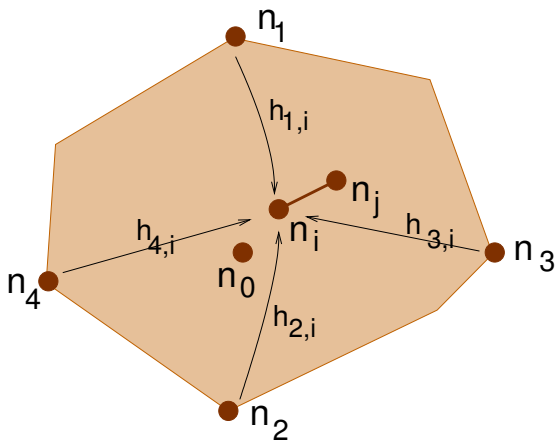


Figure 7-10: Hop count-based initial coordinate assignment in AFL phase-1.

This section describes how AFL phase-1 computes an initial coordinate assignment that approximates the physical layout of the nodes. We first assume that the graph of nodes has a fairly uniform connectivity, and that the boundary of the graph is a convex polygon (Figure 7-10). Later, we observe that AFL performs well even when these constraints are relaxed. For ease of illustration, we first describe the 2D version of the algorithm.

AFL phase-1 selects five nodes, $n_0 \ldots n_4$, as shown in Figure 7-10; these nodes are selected such that the lines joining n_1,n_2 and n_3,n_4 are roughly perpendicular to each other, and n_0 is close to the intersection of these two lines (we later discuss how these nodes are actually selected). Denote the shortest-path hop counts from nodes $n_0 \ldots n_4$ to a node n_i by $h_0(i) \ldots h_4(i)$. Treating n_0 as the origin of a coordinate system, our aim is to come up with a coordinate assignment for node i, using the values $h_0(i) \ldots h_4(i)$ and the range R, that preserves the order-correctness of a large fraction of neighboring node pairs.

We observe, for some node n_i, that the shortest-path hop count from n_0 approximately represents the ordering of n_i, with respect to its neighbors, along the line joining n_i and n_0. We also observe that $h_1(i) - h_2(i)$ approximates the "vertical" (or the y axis) displacement of node n_i with respect to its neighbors (and similarly $h_3(i) - h_4(i)$ approximates the "horizontal" displacement of n_i). Hence, the ratio

$(h_1(i) - h_2(i))/(h_3(i) - h_4(i))$ approximates the tangent of the angle $\theta(i)$ of node n_i in the polar coordinate system with the origin n_0. Based on these observations, we define the polar coordinates $(\rho(i), \theta(i))$ as follows.

$$\rho(i) = Rh_0(i) \tag{7.6}$$

$$\theta(i) = \arctan\left(\frac{h_1(i) - h_2(i)}{h_3(i) - h_4(i)}\right) \tag{7.7}$$

The equivalent Cartesian coordinates of node n_i are given by:

$$\begin{aligned} x(i) &= Rh_0(i)\frac{h_3(i) - h_4(i)}{l(i)}, l(i) \neq 0 \qquad (7.8)\\ &= 0, l(i) = 0 \\ y(i) &= Rh_0(i)\frac{h_1(i) - h_2(i)}{l(i)}, l(i) \neq 0 \qquad (7.9)\\ &= 0, l(i) = 0. \end{aligned}$$

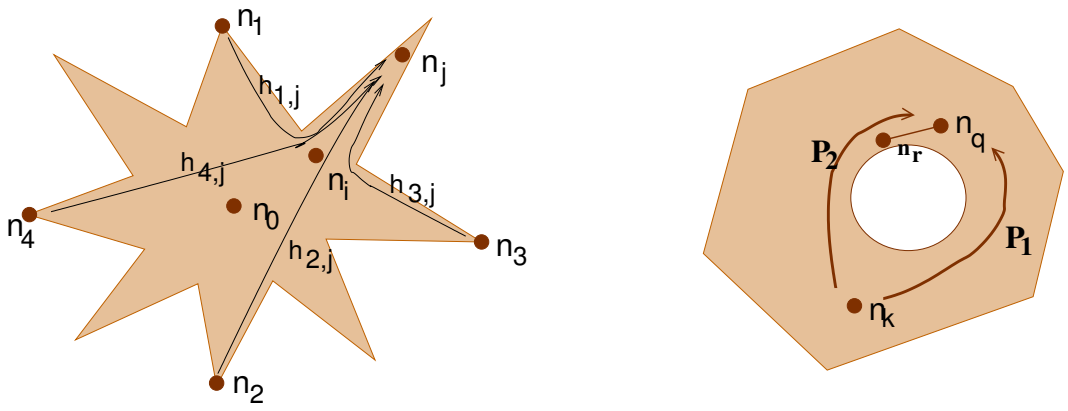
where, $l(i) = \sqrt{(h_3(i) - h_4(i))^2 + (h_1(i) - h_2(i))^2}$. Because the true distance between two neighboring nodes is always < R, representing $\rho(i)$ by $Rh_0(i)$ causes the resulting graph, G, to be an expanded version of the true node positions. Section 7.4 explains how this property helps repair order-correctness violations during the second phase of AFL.

7.3.3 AFL Phase-1 Performance Under Different Topologies

The assumption of a well-connected graph bounded by a convex polygon may not hold for some node deployments; for example, in an indoor location system like Cricket, the node deployment must follow the topology of the particular building. This section examines the AFL phase-1 performance under different graph topologies.

Consider the "star-shaped" graph in Figure 7-11. Since this graph radiates out from n_0, $Rh_0(p)$ is still a good approximation for $\rho(i)$. Consider n_i, since this is inside the "main body" of the graph, $\theta(i)$ can still be approximated by Equation (7.7). Next, consider n_j inside a "branch" of the star; we see that hop counts from nodes $n_1 \ldots n_4$ all increase approximately equally between n_i and n_j, so the θ values computed by Equation (7.7) must be approximately equal for both n_i and n_j. Hence, for star shaped graphs in general, the hop-count based initialization results in an approximately correct ordering in (ρ, θ).

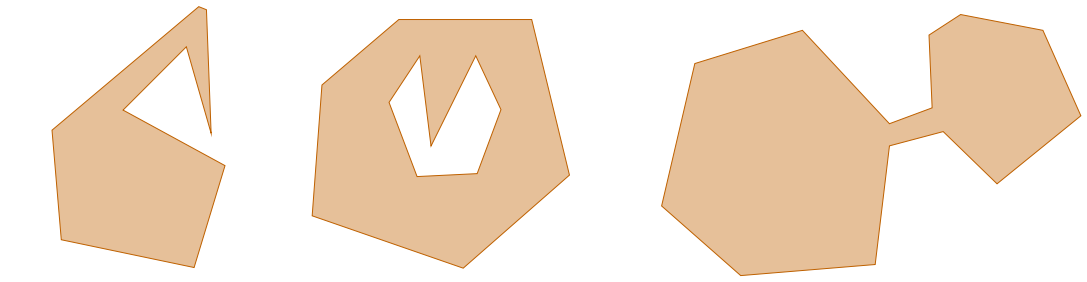
The presence of a hole (or void), as in Figure 7-11, affects the shortest-path hop-count from n_k to n_q. However, if the paths $P1$ and $P2$ have almost identical hop-counts, the relative hop counts from n_k to the neighbors of n_q continue to reflect their ordering with respect to n_k (if they are different, such that $P1 < P2$, n_q can have a smaller hop-count from n_k compared to n_r, violating the ordering). The number of order-correctness violations depends on the difference $(P2 - P1)$. Assuming that the nodes are distributed uniformly (outside the holes), and the number holes is small,



(a) Star–shaped graph (b) Graph with a hole

Figure 7-11: Initial coordinate assignment in a "star-shaped" graph and a graph with holes.

AFL phase-1 should continue to perform well in the presence of holes.



(i) Radiates inward (ii) The Hole Radiates inward (ii) Two Regions Connected by a Narrow Region

Figure 7-12: Some graph topologies that do not radiate out.

AFL performs poorly in some network topologies. For example, if the graph does not radiate outward, AFL phase-1 will have a poor order-correct performance, reaching a local minimum of $E_{SS}(G)$ during phase-2. Figure 7-12 show three different graph topologies that do not radiate outward. Phase-1 of AFL also performs poorly when the diameter (in either of the two orthogonal directions) of the graph is too small. Figure 7-13 shows three example topologies with small graph diameters.

7.3.4 Node Election in AFL Phase-1

This section describes an algorithm for electing the five nodes $n_0 \ldots n_4$ during AFL phase-1, assuming that each node has a unique ID that can be ordered. This algorithm proceeds in five steps, as described below:

- **Step 1.** Select the node n_{min} with the smallest ID. Then, select node n_1 to *maximize* $h_{min,1}$; i.e., n_1 is a node that is the maximum hop-count away from node n_{min} (Figure 7-14(a)).

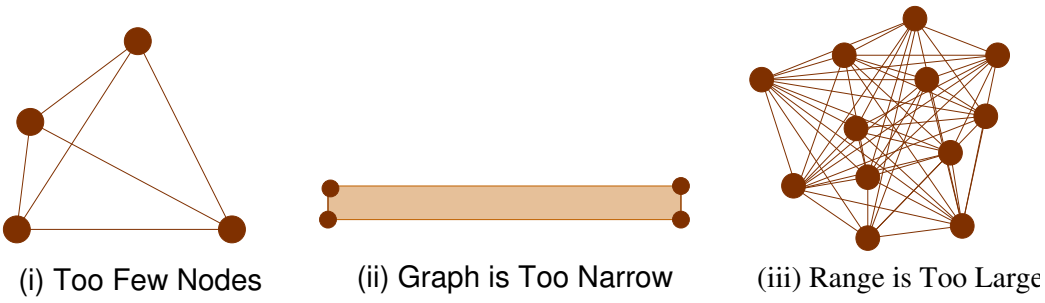


Figure 7-13: Graph topologies with limited number of hops

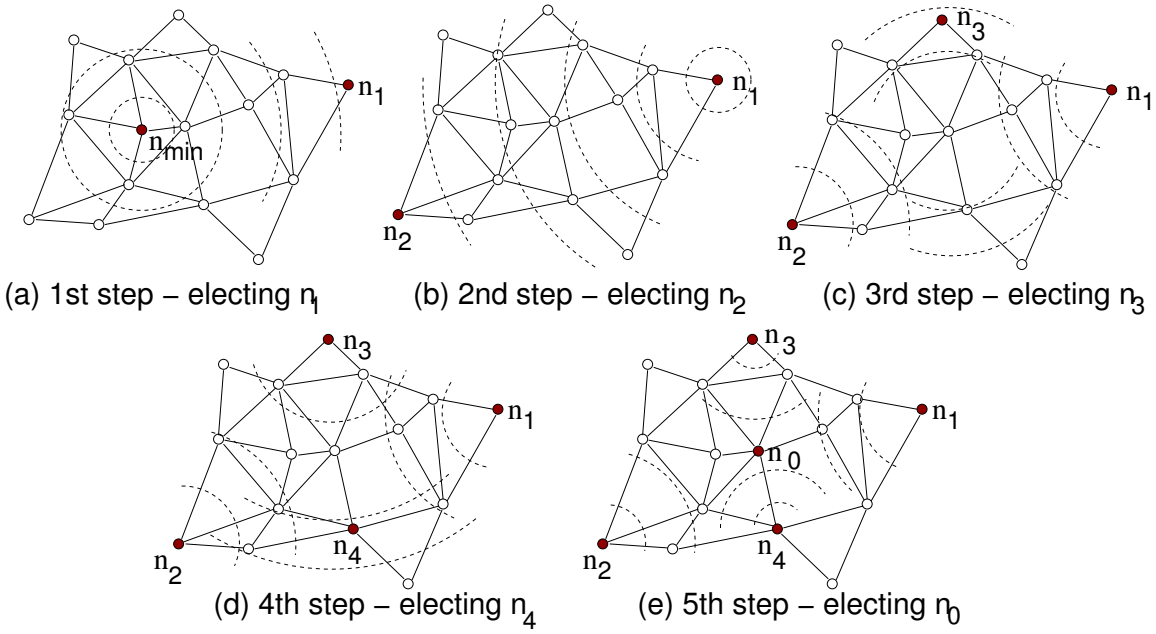


Figure 7-14: The hop-count based initialization phase of a 2D graph.

- **Step 2.** Select node n_2 to *maximize* $h_{1,2}$ (Figure 7-14(b)).
- **Step 3.** Select node n_3 to *minimize* $|h_{1,3} - h_{2,3}|$. In general, several nodes may all have the same minimum value, and the tie-breaking rule is to pick the node that *maximizes* $h_{1,3} + h_{2,3}$ from the contenders. This step selects a node that is roughly equidistant from nodes n_1 and n_2, and is "far away" from n_1 and n_2 (Figure 7-14(c)).
- **Step 4.** As in the previous step, select node n_4 to *minimize* $|h_{1,4} - h_{2,4}|$. Now, break ties differently: from among several contender nodes, pick the node that *maximizes* $h_{3,4}$. Doing so selects a node roughly equidistant from nodes n_1 and n_2, while being farthest from node n_3 (Figure 7-14(d)).
- **Step 5.** As in the previous step, select node n_0 to *minimize* $|h_{1,5} - h_{2,5}|$. From the contender nodes, pick the node that *minimizes* $|h_{3,5} - h_{4,5}|$. Doing so selects the node representing the rough "center" of the graph (Figure 7-14(e)).

For all of the above steps, ties during hop-count comparisons are broken using node IDs. After these steps, each node i calculates its initial Cartesian coordinates as described by Equation (7.8) and Equation (7.9).

7.3.5 AFL Phase-1: 3D Version

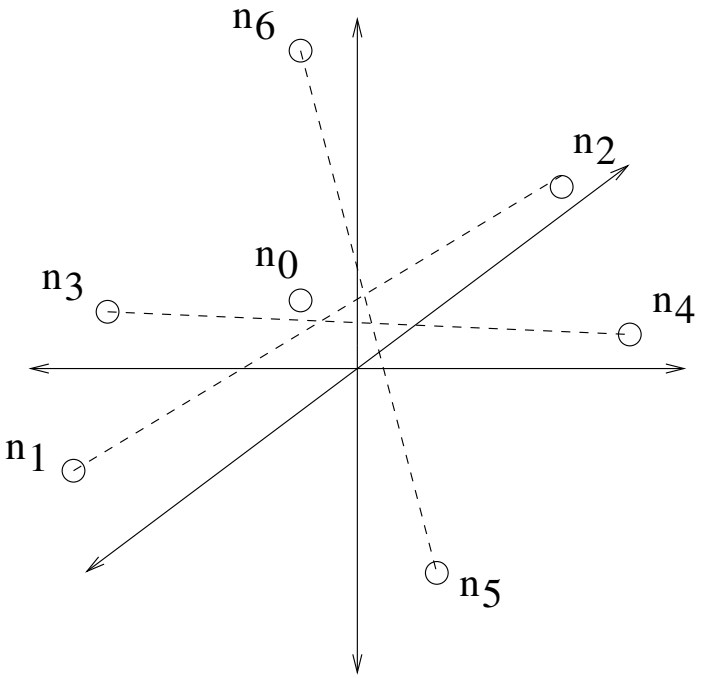


Figure 7-15: Node election for hop-count based initial coordinate assignment in a 3D graph.

The 3D version of the AFL, uses seven nodes, $n_0 \ldots n_6$, as shown in Figure 7-15 to compute the initial node coordinates. The lines joining the node pairs (n_1,n_2), (n_3,n_4), (n_5,n_6) are approximately perpendicular to each other, while n_0 is close the points of intersection of these lines. We compute the Cartesian coordinates of node i as:

$$x(i) = Rh_0(i)\frac{h_3(i) - h_4(i)}{l(i)} \tag{7.10}$$

$$y(i) = Rh_0(i)\frac{h_1(i) - h_2(i)}{l(i)} \tag{7.11}$$

$$z(i) = Rh_0(i)\frac{h_5(i) - h_6(i)}{l(i)} \tag{7.12}$$

where, $l(i) = \sqrt{(h_3(i) - h_4(i))^2 + (h_1(i) - h_2(i))^2 + (h_5(i) - h_6(i))^2}$.

Now, the node election consists of seven steps. Steps 1 through 4 are identical to the node election in the 2D version. The steps 5, 6, and 7 select nodes n_5, n_6, and n_0 as follows:

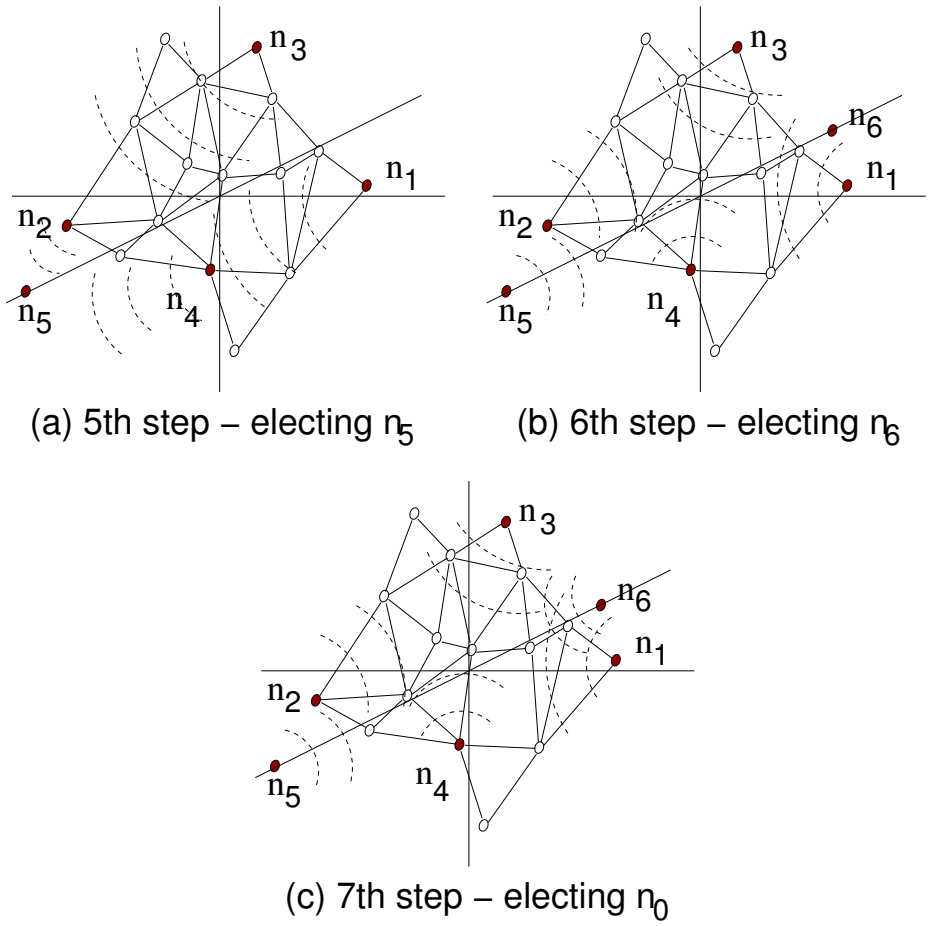


Figure 7-16: The extra steps required for AFL's hop-count based initialization phase (phase-1) for a 3D graph.

- **Step 5.** Step 5 of the 3D version elects node n_5 to *minimize* $|h_{1,5}-h_{2,5}|+|h_{3,5}-h_{4,5}|$. Break ties by selecting the node that *maximizes* $h_{1,5}+h_{2,5}+h_{3,5}+h_{4,5}$. Doing so selects a node in the "middle" of the four nodes n_1, n_2, n_3, and farthest from those four nodes (Figure 7-16(a)).

- **Step 6.** This step selects n_6, as the node that minimizes $|h_{1,6}-h_{2,6}|+|h_{3,6}-h_{4,6}|$. Use the node that *maximizes* $h_{5,6}$ to break ties. Doing so selects a node that is in the "middle" of the nodes n_1, n_2, n_3, while being farthest from n_5 (Figure7-16(b)).

- **Step 7.** This step selects the node n_0 that minimizes $|h_{1,0}-h_{2,0}|+|h_{3,0}-h_{4,0}|+|h_{5,0}-h_{6,0}|$. Doing so selects a node representing the "center" of the graph (Figure 7-16(c)).

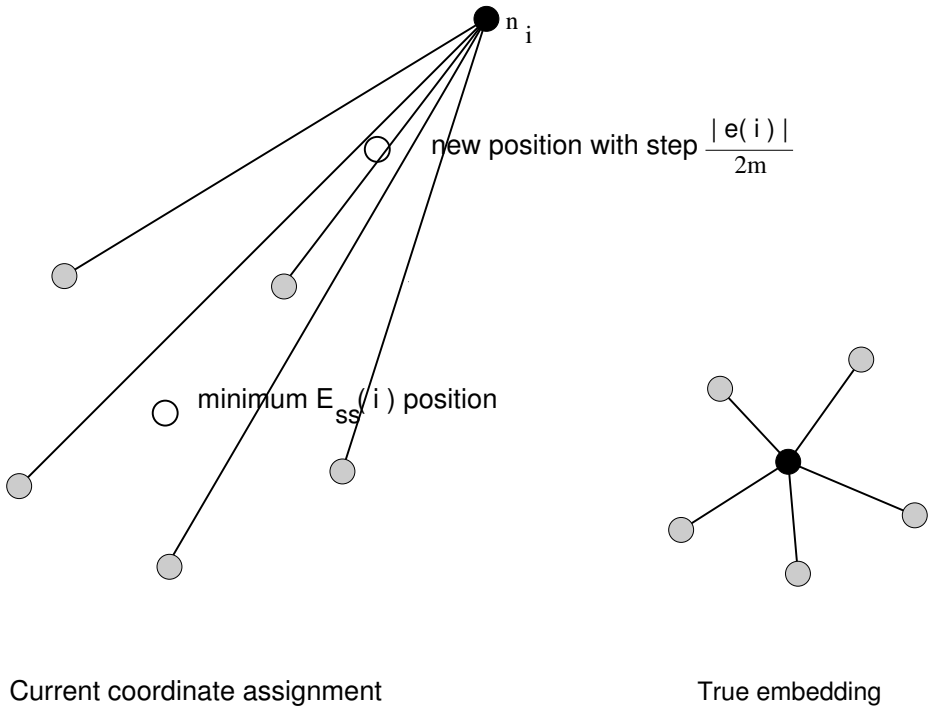


Figure 7-17: Optimization step size $\frac{|\vec{e}(i)|}{2m}$ conservatively reduces the number of order violations when the order violations are caused by a node positioned away from a set of nodes that are order-correct with respect to each other.

7.4 AFL Phase-2: Iterative Optimization

AFL phase-2 minimizes the sum-squared-error $E_{ss}(G)$ of the graph by iteratively reducing the sum-squared-error $E_{ss}(i)$ of individual nodes. AFL phase-1 results in an approximately order-correct graph. Hence, according to proposition 7.1 after AFL phase-1, it is likely that there is some node n_i with non-zero resultant error. Theorem 7.1 tells us that we can, in general, update the position of n_i along its resultant error to reduce $E_{ss}(i)$, and hence reduce $E_{ss}(G)$. However, proposition 7.1 further tells us that we should continue to preserve the order-correctness while updating the node positions, so that we have a better chance of reaching the global minimum by iteratively updating the positions of all the nodes. Hence, a greedy algorithm that minimizes the energy of a node under given configuration by searching for the minimum $E_{ss}(i)$ along $\vec{e}(i)$ may not work, because it may reduce the fraction of correctly ordered edges. We need to determine an appropriate step size for the node position update that tends to reinforce the order-correctness of the graph (the update step size is an important parameter in non-linear optimization in general [9]).

We decide on an appropriate step size based on the following observation. Figure 7-17 shows a possible hypothetical order violation. Here, all the neighbors of node n_i are in the proper order with respect to each other, but node n_i is positioned far from its neighbors, causing order violations with respect to its neighbors. Since AFL phase-1 results in a coordinate assignment with a small fraction of order violations,

```
run_phase_1_and_compute_initial_coordinates();
while true do
   /* wait for slow neighbors */
   repeat
      if a neighbor timed out, mark as dead;
   until min_neighbor_step < my_step;
   compute p⃗ = e⃗/2m;
   /* linear search */
   for  i=1 to  10 do
      if Ess(current_pos) < Ess(current_pos + p) then
         est_pos = est_pos + p;
         break;
      endif
      else
         p = p*0.8;
      endif
   endfor
   my_step = my_step + 1;
endw
```

Figure 7-18: AFL pseudo code (distributed version).

assuming that these violations are distributed uniformly over the whole graph, the above hypothetical situation suggests the type of order violations we can expect after AFL phase-1.

The coordinate assignment of phase-1 results in an expanded version of the graph. Hence, the area of the polygon p, comprising the neighbors of node n_i, after phase-1, should be larger than the area of that polygon in the true layout. Under these conditions, the minimum energy position of node n_i would fall within somewhere in the middle of p as shown in Figure 7-17. This minimum energy position seems to correspond to a small number of order violations compared to the current position of n_i. When the offset between the middle of p and the current position of n_i is large, an update step size of $\frac{|\vec{e}(i)|}{m}$, in the direction of $\vec{e}(i)$, places n_i at a point close to the middle of p. However, since this update step size could increase the number of order violations due to n_i over shooting, we select a more conservative position update scheme with a maximum update step size of $\frac{|\vec{e}(i)|}{2m}$ where m is the number of neighbors of n_i. This update step size places n_i at a point located between its current position and its minimum energy position, thus preventing the possibility of more order violations. In AFL phase-2, each node n_i computes the maximum step size $\frac{|\vec{e}(i)|}{2m}$, and then performs a binary search to select an update step size between $(0, \frac{|\vec{e}(i)|}{2m})$ that minimizes $E_{ss}(i)$ along $\vec{e}(i)$. In section 7.6 we use simulations to show that this step size actually reduces the number of order violations during AFL phase-2.

We note that phase-1 and phase-2 of AFL can be implemented as distributed algorithms.

When AFL phase-2 runs as a distributed algorithm, different nodes may run

at different speeds. This speed variation can have an adverse effect on distributed optimization algorithms in general [85]. In AFL, nodes updating at different speeds can increase the number of order-correctness violations, resulting in $E_{ss}(G)$ reaching a local minimum. We use the following approach to prevent different nodes from executing AFL phase-2 at widely varying speeds. Each node n_i keeps track of the number of optimization steps, s_i, it has performed after the initialization phase. After each optimization step, each node n_i shares its current coordinates and the value s_i with its neighbors. Each node n_i waits until $s_i \leq s_j$ for every n_j if $i \leftrightarrow j$, before the next optimization step; thus AFL allows at most one optimization-step offset between neighboring nodes.

Figure 7-18 gives the pseudo-code of the distributed version of AFL. All the nodes run this algorithm concurrently, after some node broadcasts a start message. Here, *min_neighbor_step* is the minimum step value heard from all the neighbors, and E_{ss} is the sum-squared error of the particular node.

7.5 Using MAT with AFL

Once we obtain the inter-beacon distances using one of the approaches described in section 6.3, we can run the AFL algorithm to compute a coordinate assignment for the Cricket beacons. However, there are two important aspects that we need to pay attention to when using the combination of MAT and AFL to localize Cricket beacons deployed in a typical indoor environment. First, the graph of nodes obtained from MAT may not have the proper structural requirements to run AFL phase-1. Second, the deployment of beacons may not represent a true 3D or 2D structure, which can cause complications during AFL phase-2. We examine these issues and discuss possible solutions below.

7.5.1 AFL Initialization using RF Connectivity

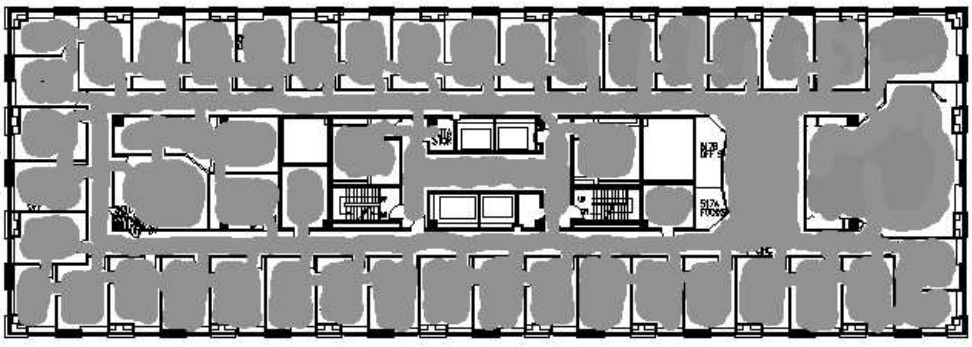


Figure 7-19: The structure of the graph of beacons in an example indoor environment.

The performance of AFL phase-1 depends on the assumption of a well-connected graph with only a limited number of voids in the graph. Since the MAT algorithm

described in the previous section uses the distances collected at a mobile listener to obtain inter-beacon distances to build a graph of beacons, the shape of the resulting graph is similar to a collection of rooms that are interconnected by a collection of doorways and corridors. For example, Figure 7-19 show a graph that spans a collection of rooms that are interconnected by doorways and corridors. We observe that this graph does not have the proper structural properties required by AFL phase-1. However, unlike inter-beacon estimates from MAT, RF signals from Cricket beacons can penetrate boundaries such as walls; hence, a graph based on the RF connectivity between Cricket beacons is likely to have a more uniform connectivity, resulting in a structure amenable for running AFL phase-1. Although RF connectivity is a poor measure of beacon proximity, as Section 7.3.1 showed, the tolerance of the shortest-path hop count-based node ordering to variations in the range enables us to use the RF connectivity between beacons to obtain an initial coordinate assignment that approximates the physical layout of the beacons.

7.5.2 Dealing with Semi-3D Beacon Deployments

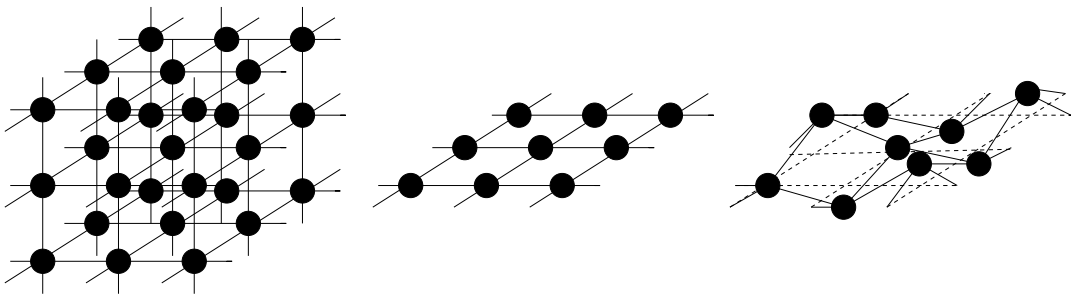


Figure 7-20: An example showing how semi-3D structures differ from 3D and 2D structures.

A typical indoor deployment of beacons, e.g., on a single floor of a building, will have a "semi-3D" structure as opposed to a "true" 2D or 3D structure. A "semi-3D" structure is one whose third dimension is small compared to the other two; for example, a crumpled piece of paper forms a "semi-3D" structure. An indoor beacon deployment with line-of-sight ranging will not be true 2D because beacons will generally be displaced vertically depending on coverage requirements and availability of mounting points. It will not be true 3D because, for a single floor of a building, there is only one "layer" of beacons covering the floor (when beacons are deployed across multiple floors, connectivity across floors would usually happen through narrow stairwells, making it difficult to generate rigid interconnections across floors).

The semi-3D nature of an indoor beacon deployment creates difficulties for localization schemes that use optimization to determine node positions. Because a small error in the vertical direction during the initialization phase can place a node on the wrong side of the polygon created by its neighbors, that node may get trapped in a

potential well during the optimization. It is difficult to bring these nodes out of the potential wells using trial-and-error because of the large state space of this approach.

When localizing a semi-3D beacon deployment using AFL based on the inter-beacon distances obtained in MAT, we slightly modify AFL to take advantage of additional information available from MAT. For example, when we compute inter-beacon distance using three beacons at a time (Section 6.3), while moving the mobile node on a plane parallel to ground, the z values of the computed local coordinates of the beacons give the heights of individual beacons above this plane. If the listener moved in the same plane while collecting distances from all the beacons, then the z values give the height of the beacons above the plane containing the listener.

We modify AFL to use this additional information as follows. First we run the AFL initialization for 2D graphs, which results in a graph that is approximately order-correct in 2D space. Next we assign each beacon z coordinates based on the z values we obtained from MAT. During AFL phase-2, we run the optimization algorithm only in the $X - Y$ plane by setting the z components of node updates to zero. Because z coordinates are known to be correct, the optimization phase will assign proper (x, y) coordinates to individual nodes. This approach has the additional advantage that we need only a smaller number of inter-beacon distances compared to the full 3D version, because the graph only need have 2D rigidity instead of 3D rigidity.

There are instances where the assumption of moving the mobile in a horizontal plane may fail. For instance, if the floor is not perfectly flat, the mobile will not be able to move in the same horizontal plane. Similarly, there may be vertical jitter of the mobile position when it is carried around by a user to collect distances. In these situations, although the z values from MAT may no longer represent the height of beacons from a plane parallel to ground, these z values may represent the correct vertical ordering of the beacons, assuming that the vertical displacement of beacons is larger than the vertical jitter of the mobile. We can use this ordering information of beacons in the z direction to prevent beacons getting trapped in potential wells due to movements in the z direction.

7.6 Simulation Results

We ran simulations to evaluate different aspects of the AFL algorithm. We wrote a Java3D-based simulator that enabled us to analyze, and visualize the performance of AFL. All the results presented here are for 2D simulations. When deploying nodes in these simulations, we maintain a uniform local density by adding nodes to only those regions that have a number of nodes below a certain threshold (we select this threshold based on the average graph-connectivity required).

7.6.1 Importance of Order-Correctness for Minimizing Graph Energy

In this experiment, we verified the importance of the order-correctness for the sum-squared error E_{ss} to reach the global minimum during the optimization. We selected

a graph G of n nodes, and initialized the coordinates of each node by selecting a random point, with a uniform distribution, within the area covered by the true node deployment. We then ran the optimization phase of AFL on G, and checked if we reach the global minimum.

We simulated graphs of 30, 100, and 300 nodes, with average per-node connectivities of 4, 8, and 12 for each case. For each combination of graph size and connectivity, we ran 20 simulations. *All of these* $3 \cdot 4 \cdot 20 = 240$ *simulations resulted in local minima.* These results demonstrate that an arbitrary initialization followed by the minimum energy relaxation performs poorly when trying to reach the global minimum.

7.6.2 AFL Phase-1 Performance

We used these simulations to determine the effectiveness of the hop-count based initialization, by observing the fraction of order-correctness violations. We also examined how network connectivity affect these order-correctness violations. We simulated 20 different instances of 400-node graphs of a given average connectivity (node degree). After running AFL phase-1, we calculated the ρ and θ (in polar coordinates) order-correctness violations. Figures 7-21 7-22 show the results of these simulations (error bars show the standard deviation).

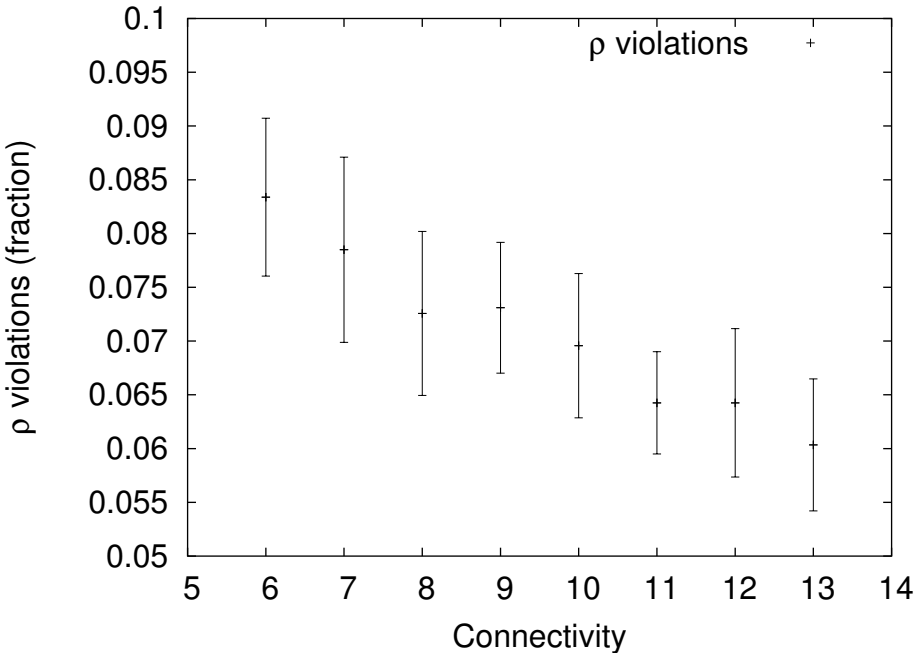


Figure 7-21: The fraction of ρ violations after AFL phase-1 vs. connectivity.

We observe that the initialization phase of AFL has only a small-fraction of order-correctness violations, and that the fraction of violations decreases with increasing connectivity.

During the discussion of AFL, we assumed the "range" R to be a constant for a given graph, but in practice R can have some variations (for example, when we use

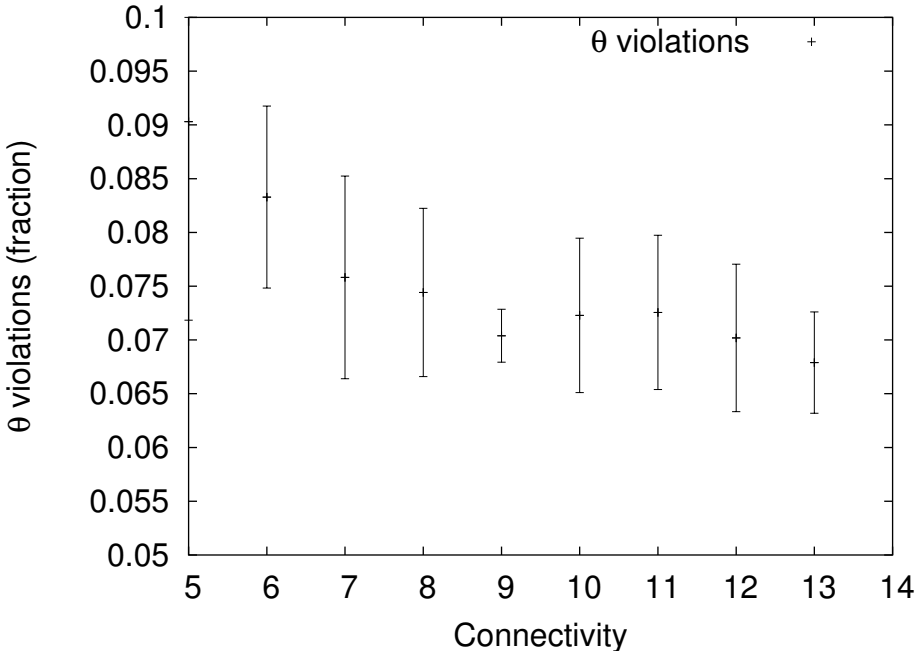


Figure 7-22: The fraction of θ violations after AFL phase-1 vs. connectivity.

RF connectivity to determine neighbors). The following simulations examined how variations in the value of R affect the AFL phase-1 performance. We used 400-node graphs with different connectivities (6 to 12) and different percentage variations in R. For each percentage variation in R and graph connectivity, we simulated 20 graph instances. We assumed that different values of R are uniformly distributed within the particular range. Figure 7-23 plots the order-correctness violations for different percentage variations of R. We observe that while order-correctness violations increase with the percentage variation of R, the fraction of order-correctness violations remains small. This shows that the initial phase of AFL is robust to variations in R.

7.6.3 AFL Phase-2 Performance

We discussed how the "scaled-up" initial configuration and proper optimization step size reinforces the order-correctness of the graph during the AFL optimization phase. We deployed graphs of 400 nodes each, with connectivities from 6 to 12 in increments of 2. For each connectivity, we ran 10 simulations. During the optimization phase of AFL, we calculated the percentage of ρ and θ violations after each optimization step. The graph in Figure 7-24 shows the variation of the order-correctness violations (both ρ and θ) as a function of the (average) number of optimization steps per node. Within 15 iterations, the fraction of violations drops to about half, and the order-correctness violations continue to decrease as the algorithm progresses.

We next examined the running time of the optimization phase. We used n nodes with an average connectivity of 8. We ran AFL on these graphs and obtained the number of optimization steps required for the average percentage edge length error

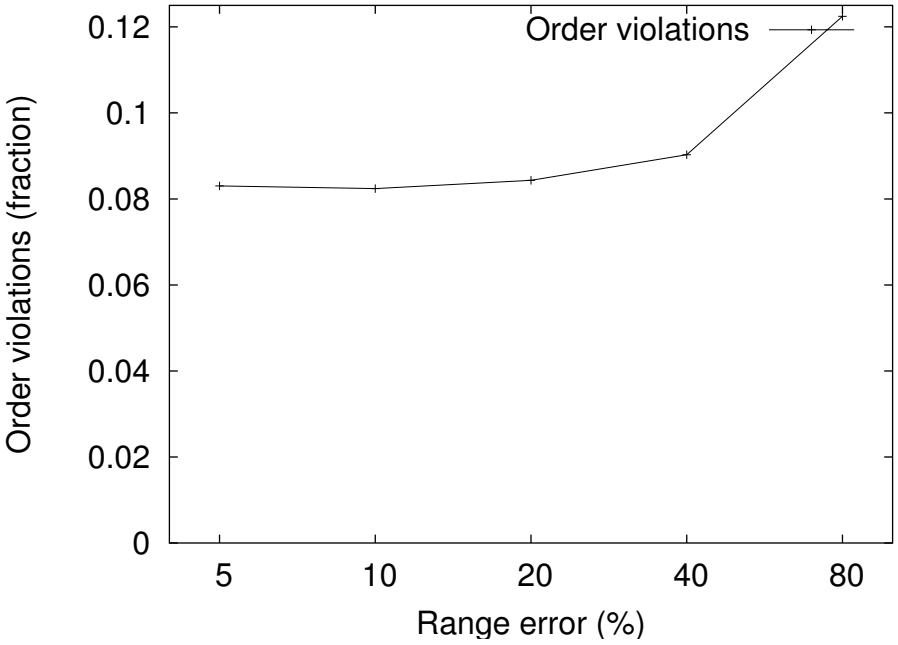


Figure 7-23: The variation of the fraction of violations vs. percentage variation in R.

to reach 1%. We varied n from 100 to 1000 in steps of 100. We ran 10 simulations for each value of n. Figure 7-25 shows the variation of running time (optimization steps) of the optimization phase (per node) as a function of the number of nodes in the graph. Figure 7-26 shows the variation of running time as a function of the graph diameter.

Next, we evaluate the effectiveness of our solution for nodes proceeding at different speeds during the optimization phase. High-speed nodes can perform optimization steps 10 times faster than conventional nodes. We experimented with different ratios of high-speed and conventional nodes. The first two scenarios used 10% and 20% present of high-speed nodes distributed uniformly with conventional nodes. The third scenario deployed high-speed nodes in a contiguous region covering a quarter of the graph. For comparison, the fourth scenario used no-high speed nodes. For all four cases, we ran 500-node simulations for connectivities from 6 to 12 in increments of 2. For a given connectivity and high-speed node deployment, we ran 50 simulations. We did not observe any significant difference in performance regarding the ability to reach the global minimum of the graph energy $E_{ss}(g)$ among these four cases. This shows that the "simulation-step" count approach prevents problems due to nodes running at different speeds.

7.6.4 Overall performance of AFL

In this set of experiments, we examined the performance of AFL algorithm as a whole. In our first experiment, we examined the performance under different network connectivities. For each value of network connectivity, we run AFL on 50 instances of

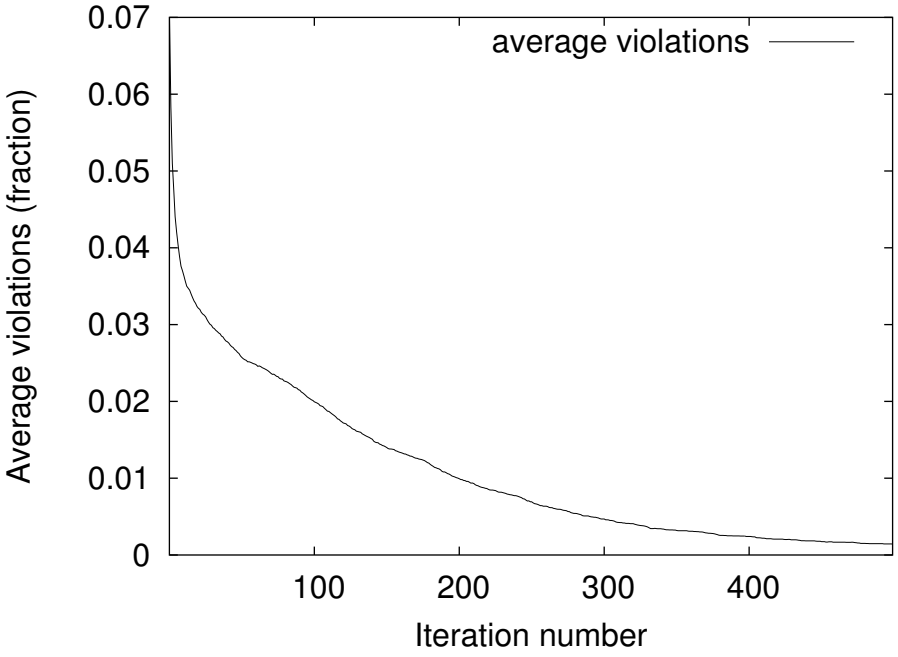


Figure 7-24: The variation of the fraction of violations during optimization.

250-node graphs. Figure 7-27 gives the results of this experiment. For connectivities above 7, AFL manages to localize the graph almost 100% of the time. Even for very low connectivities, AFL performs reasonably well.

Because all practical ranging technologies have some measurement error, AFL needs to be robust against such errors. Our next experiment examines the performance of AFL under distance measurement errors. Here we model errors by a uniform distribution. We simulate 250-node graphs, and each point on the graph represents 50 simulations. Different points on the graph represent different combinations of connectivity and distance measurement error.

Figure 7-28 shows the average error between all the node-pairs in the graph. We use the error between all the node-pairs, as opposed to between pairs with measured distances, to capture the global properties of the resulting graph. As expected, the % unconnected error increases with measurement error, while it decreases with increasing connectivity, since increased connectivity improves the performance of the optimization phase. According to the graph, we observe that AFL performs well under measurement errors.

7.7 AFL Experimental Results

We ran AFL on the Cricket beacon deployment described in the Section 6.6 (Figure 7-29). To run AFL phase-1, we obtained the RF connectivity between beacons as follows. Each beacon transmits an `RF_CONNECTIVITY` message. Each beacon keeps track of the number of `RF_CONNECTIVITY` messages it receives from other beacons,

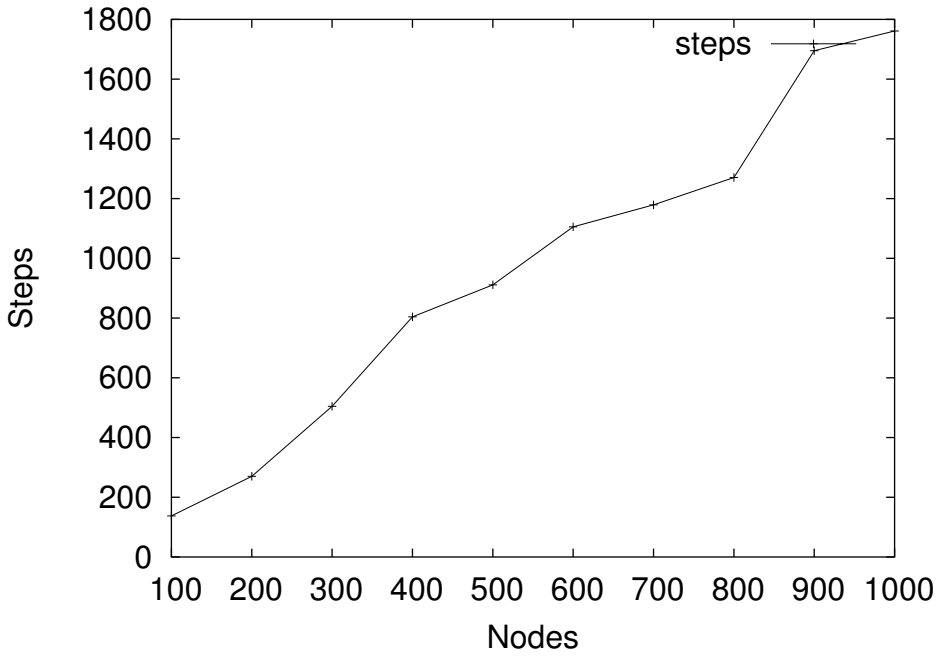


Figure 7-25: The running time of the optimization phase as a function of the number of nodes.

and it times out the per beacon message counts using an expiration timer with an expiration time inversely proportional to the total message arrival rate. This approach filters out infrequent messages from far-away beacons, and the beacons with a high message arrival rate are considered neighbors.

Figure 7-30 shows the layout resulting from the initial coordinate assignment obtained by running AFL phase-1 on the graph defined by the RF connectivity. Although this is a poor representation of the original node deployment shown in Figure 7-29, we find that nodes are sufficiently well distributed to prevent AFL phase-2 from reaching a local minimum. We also note that the performance of RF connectivity-based initial coordinate assignment improves as the geographic region covering the beacon deployment becomes larger compared to the typical RF range.

Using the beacon coordinates in the Figure 7-30 as initial coordinates, we ran the AFL phase-2 on the set of beacons. For AFL phase-2, we used the graph and the inter-beacon distances we obtained during MAT (Figures 6-13 6-14).

Figure 7-31 shows the beacon coordinates obtained after running AFL phase-2 and the true beacon locations after translating and rotating the two graphs to obtain the minimum sum squared distance between the node positions in the two graphs. We observe that the error between the estimated and true node positions is small, comparable to the errors from the MAT algorithm (Figure 6-14). This demonstrates that the combination of MAT and AFL enables us to compute beacon coordinates in a typical indoor beacon deployment.

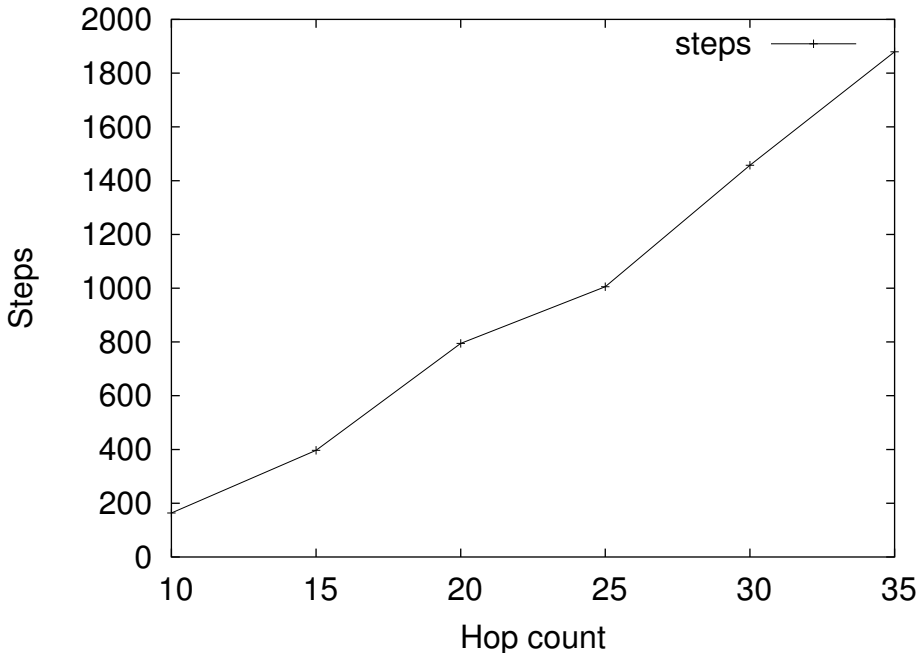


Figure 7-26: The running time of the optimization phase as a function of the graph diameter.

7.8 Chapter Summary

This chapter described the AFL algorithm for computing a beacon coordinate assignment using inter-beacon distances and RF connectivity between beacons. This chapter described two theorems that guide the AFL algorithm. AFL phase-1 elects a number of nodes and uses hopcounts from these nodes to compute an initial coordinate assignment that approximates a scaled up version of the beacon layout. AFL phase-2 uses inter-beacon distance estimates, obtained from MAT, to iteratively improve the beacon coordinate assignment. This chapter used simulation and experimental results to examine the performance of the AFL algorithm.

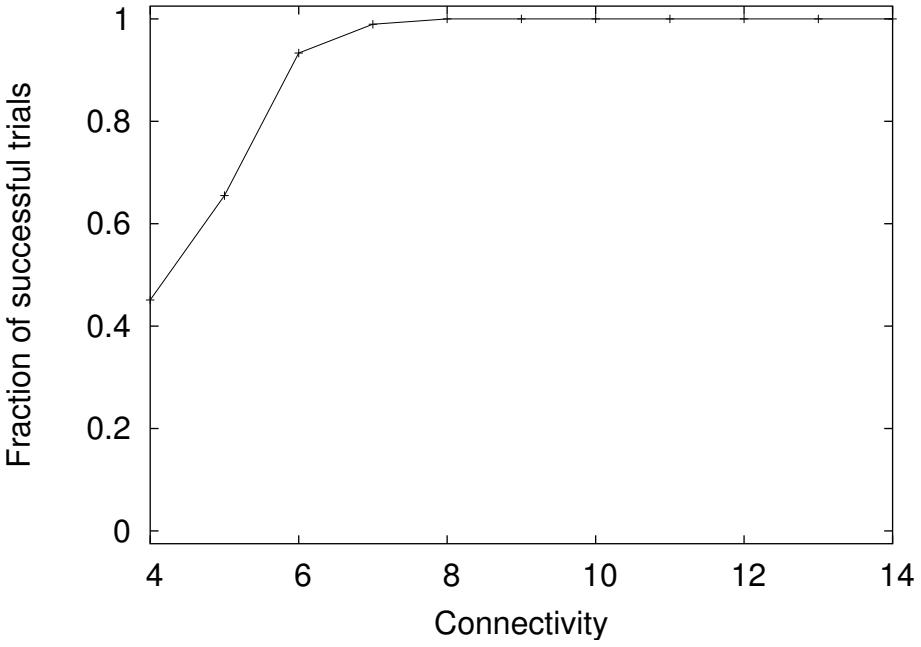


Figure 7-27: The fraction of the time AFL managed to localize a graph.

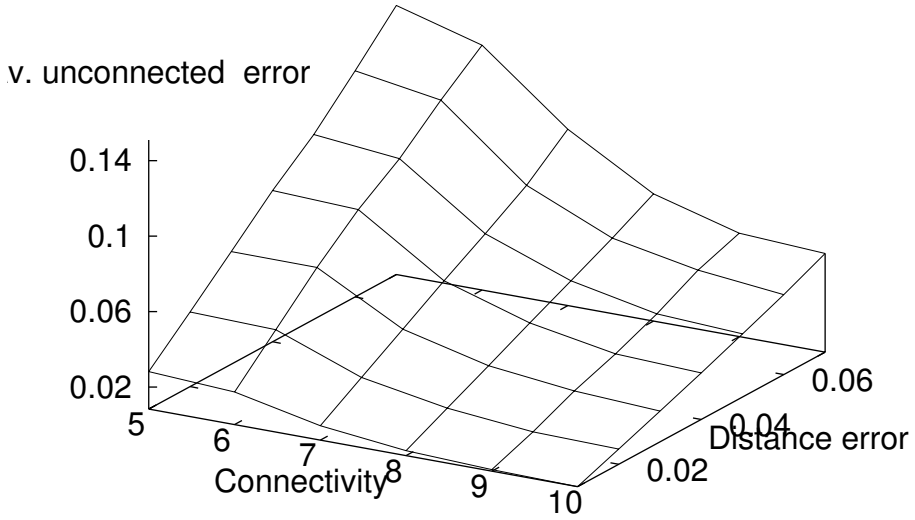


Figure 7-28: Average error between unconnected nodes.

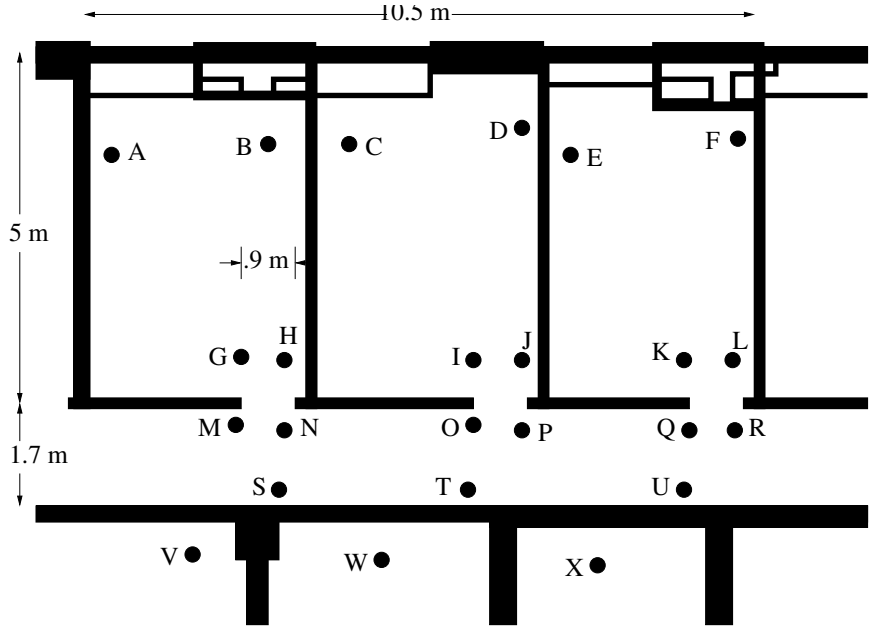


Figure 7-29: An indoor deployment of 24 nodes to evaluate the performance of MAT and AFL (reproduced).

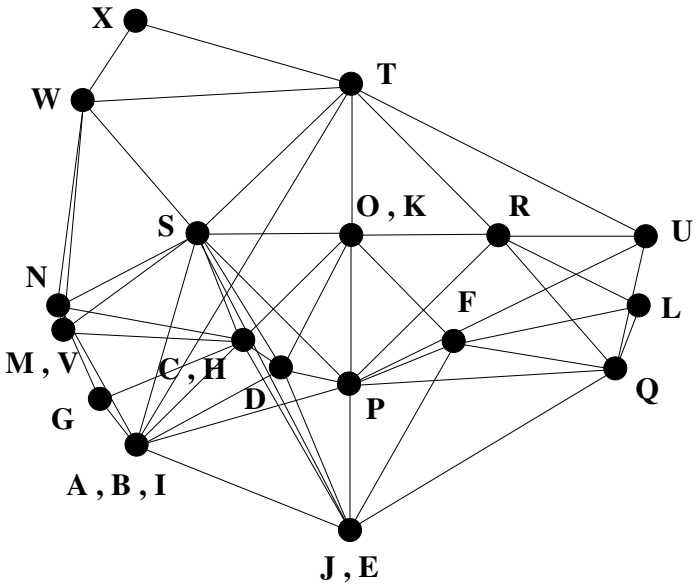


Figure 7-30: Graph obtained after running the AFL initialization.

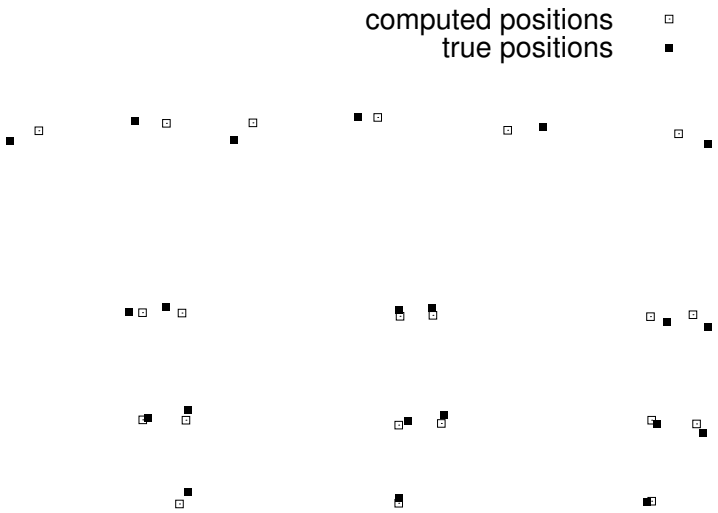


Figure 7-31: Coordinates obtained after running the AFL optimization, in comparison with the original node positions. The left-to-right distance between the furthest nodes is about 10 meters.

Chapter 8

Conclusion

This dissertation described the design, implementation, and evaluation of the Cricket indoor location system. Cricket is an easy to deploy, scalable, and accurate location system that provides user location without compromising user privacy.

This chapter summarizes the challenges encountered when implementing an indoor location system like Cricket, and the contributions made in this dissertation. We also discuss possible future directions for improving and extending the functionality of Cricket.

8.1 Challenges

There are four main challenges in developing an indoor location system:

- **Accuracy:** Many indoor applications require a position accuracy of a few centimeters and an orientation accuracy of a few degrees. The harshness of indoor environments on signal propagation, caused by obstacles, makes it hard to achieve these accuracies. It is also necessary to provide different types of location information to support diverse indoor applications—user space(which requires accurate boundary detection), position in a coordinate system, and orientation.

- **Scalability:** Indoor environments often contain a large number of physical objects and a large density of people, all requiring location. Hence, an indoor location system needs to scale well with the number and the density of users of the system.

- **User privacy:** The ability to obtain user location without the location system tracking the current location of the user is important to build applications that preserve user privacy.

- **Ease of deployment:** The location system should be easy to deploy, configure, and maintain. The amount of manual configuration and precise placement should be as small as possible.

8.2 Contributions

This dissertation describes the design, implementation, and evaluation of Cricket. Cricket consists of beacons that are attached to the ceiling of a building, and listeners attached to various devices. Each beacon periodically transmits its location information in an RF message, at the start of this message, the beacon also transmits an ultrasonic pulse. The listeners listen to beacon transmissions and measure distances to nearby beacons using the arrival times of the RF and ultrasonic signals. Listeners use these distances to compute their space, position, and orientation.

This dissertation makes the following contributions, while meeting the above challenges.

- **Hardware implementation:** The Cricket hardware uses a combination of RF and ultrasound to measure beacon-to-listener distances accurately. The Cricket compass hardware, with a unique arrangement of sensors, enables accurate orientation estimation by estimating differential distances precisely.

- **Interference avoidance and detection:** The interference avoidance and detection algorithms used in Cricket enable a distributed system architecture with uncoordinated beacon transmissions. No wiring or centralized scheduling of the beacons is necessary.

- **Mobile assisted topology generation (MAT):** MAT uses measurements taken at a mobile listener to compute the inter-beacon distances, resulting in a rigid structure of beacons for computing beacon coordinates.

- **Anchor free localization (AFL):** AFL uses inter-beacon distances estimated during MAT to compute a coordinate assignment that reflects the beacon layout. AFL runs in two phases. First it uses RF connectivity between beacons to compute an initial coordinate assignment that results in an unfolded and scaled up version of the beacon layout. The second phase of AFL uses inter-beacon distance estimates from MAT to iteratively improve the current beacon coordinate assignment. The combination of MAT and AFL eases the deployment and configuration of the Cricket system.

The Cricket hardware implementation is commercially available from Crossbow Technologies (`http://www.xbow.com`). The Cricket software implementation, documentation (including a user manual), and hardware design are available from `http://cricket.csail.mit.edu`.

8.3 Future Directions

Although the inter-beacon interference avoidance algorithms used in Cricket can detect and filter out interference, the interfering transmissions are wasted since since they are removed. Disambiguating these interfering signals, for example using modulated ultrasonic signals, rather than filtering them out, will improve performance.

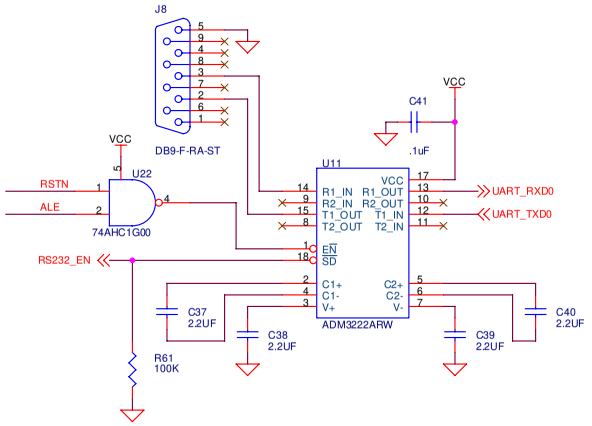
Because of the occasional lack of line-of-sight connectivity to nearby beacons and low beacon densities, it is difficult to deploy beacons to provide 100% coverage within an environment. A multi-modal approach that includes other types of sensors such as accelerometers, GPS receivers, RF-ID readers, etc. may be better suited to provide truly ubiquitous location information to a mobile user.

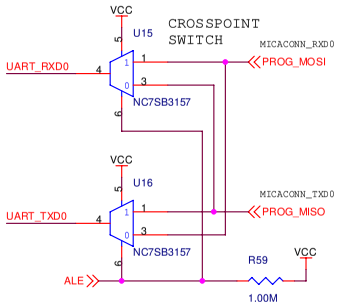
The combination of MAT and AFL uses measurements collected at a mobile listener to configure the beacons with a coordinate assignment that reflects the layout of the beacons. We can extend this to solve the general Simultaneous Localization and Mapping (SLAM) problem that involves the simultaneous implementation of a location system and the generation of a map of the region. During MAT, the mobile can use a scanning laser range finder to find the distances to walls and other obstacles in the environment. After AFL, we can obtain the locations at which the mobile node collected distance samples. From these mobile positions and measurements from the laser range finder, we can compute the location of the walls and other obstacles, thus generating a map of the environment.

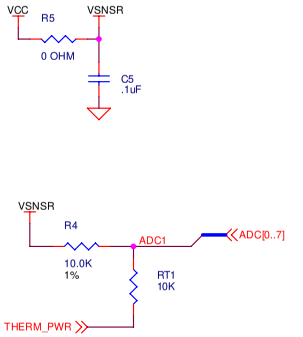
Some other potential improvements are to replace the ultrasound-based ranging with RF-based ranging technologies such as UWB which will not require as many "beacons". Another potential improvement is to use a small set of tightly coordinated beacons to improve the tracking behavior of Cricket for applications such as game consoles.

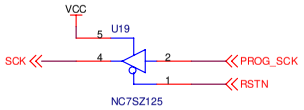
Appendix A

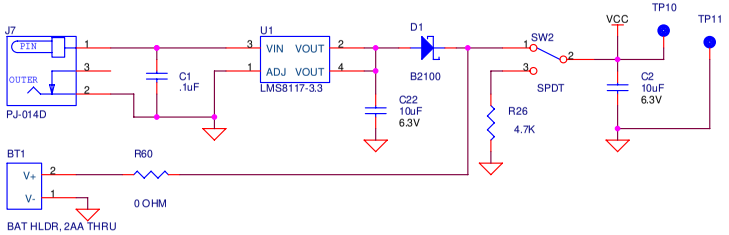
Cricket Schematic





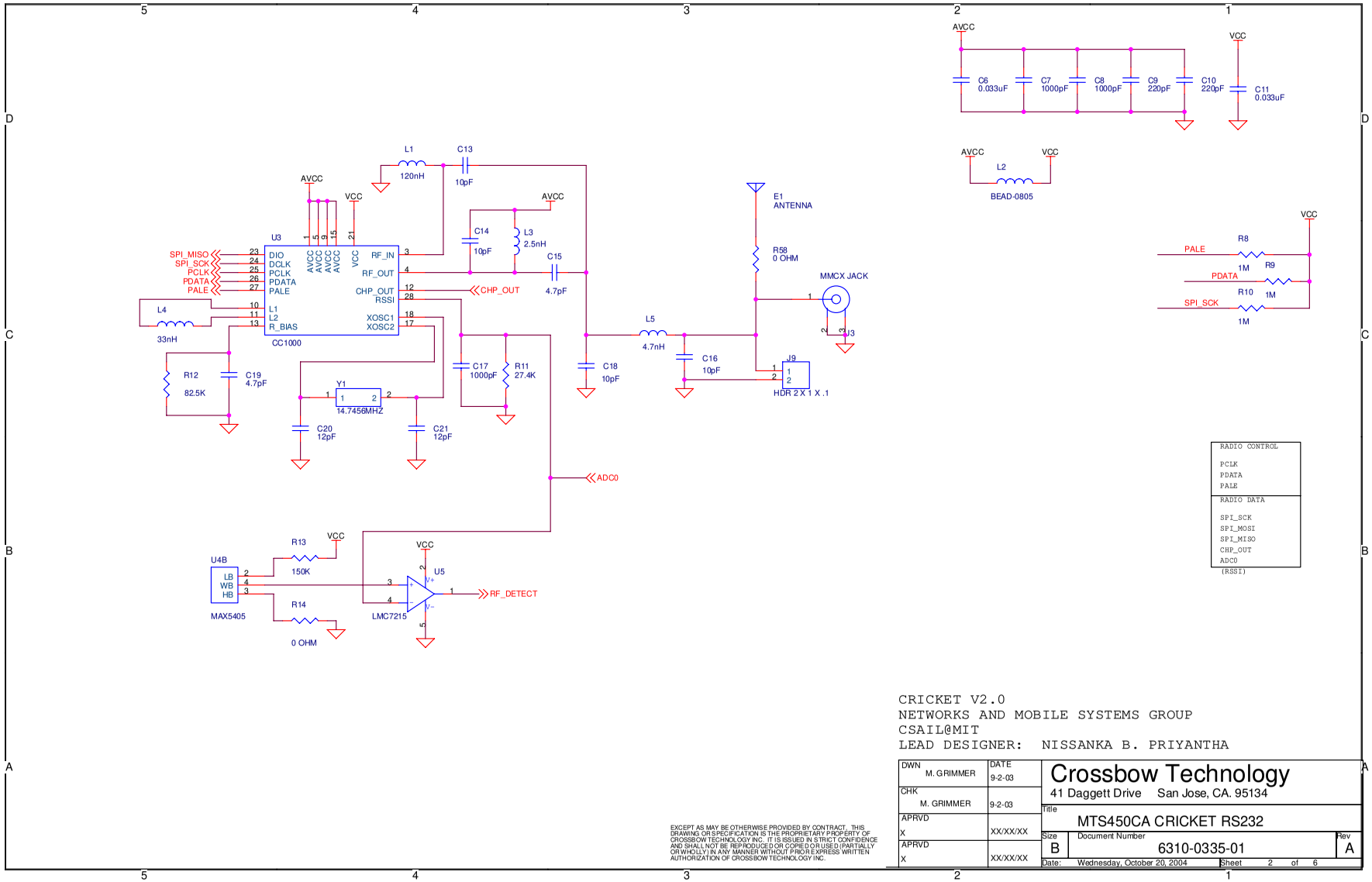






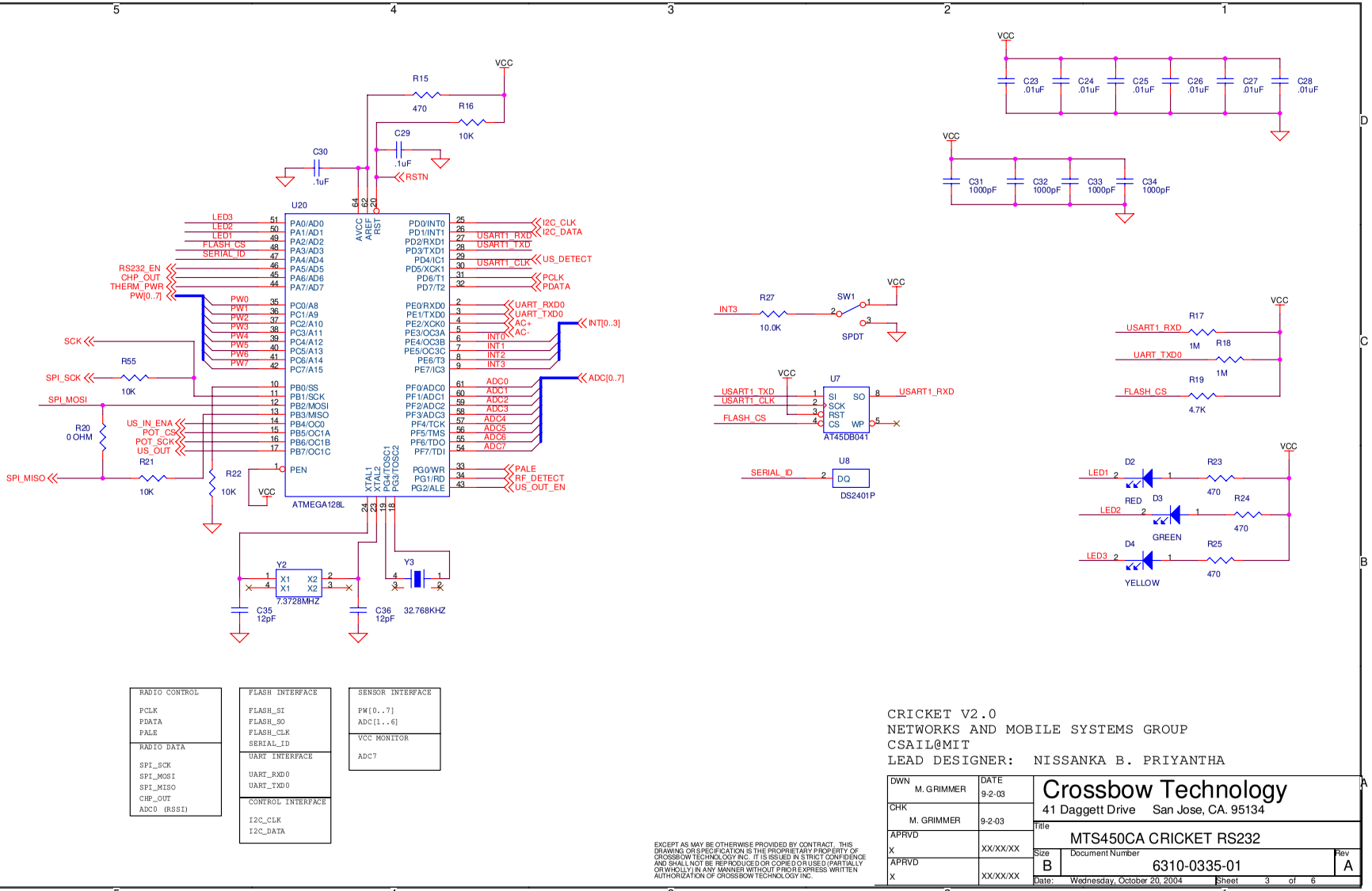
CRICKET V2.0
NETWORKS AND MOBILE SYSTEMS GROUP
CSAIL@MIT
LEAD DESIGNER: NISSANKA B. PRIYANTHA

DWN M. GRIMMER	DATE 9-2-03	Crossbow Technology
CHK M. GRIMMER	9-2-03	41 Daggett Drive San Jose, CA. 95134
APRVD X	XX/XX/XX	Title: MTS450CA CRICKET RS232
APRVD X	XX/XX/XX	Size B; Document Number 6310-0335-01; Rev A
		Date: Wednesday, October 20, 2004; Sheet 1 of 6



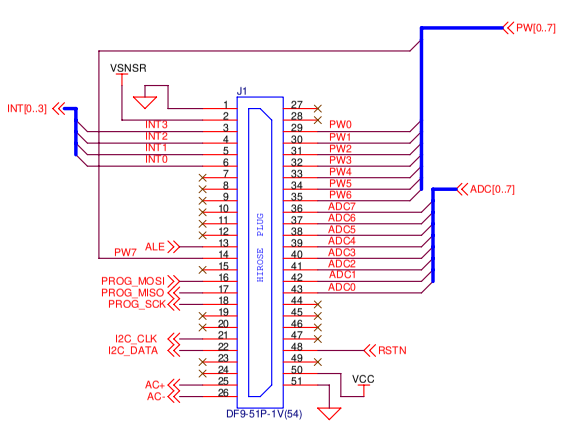
CRICKET V2.0
NETWORKS AND MOBILE SYSTEMS GROUP
CSAIL@MIT
LEAD DESIGNER: NISSANKA B. PRIYANTHA

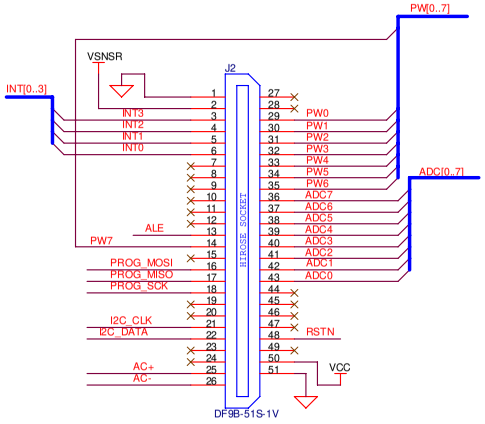
DWN M. GRIMMER	DATE 9-2-03	Crossbow Technology	
CHK M. GRIMMER	9-2-03	41 Daggett Drive San Jose, CA. 95134	
APRVD X	XX/XX/XX	Title: MTS450CA CRICKET RS232	
APRVD X	XX/XX/XX	Size: B	Document Number: 6310-0335-01 Rev: A
		Date: Wednesday, October 20, 2004	Sheet 2 of 6

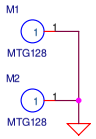


CRICKET V2.0
NETWORKS AND MOBILE SYSTEMS GROUP
CSAIL@MIT
LEAD DESIGNER: NISSANKA B. PRIYANTHA

DWN M. GRIMMER	DATE 9-2-03	Crossbow Technology	
CHK M. GRIMMER	9-2-03	41 Daggett Drive San Jose, CA. 95134	
APRVD X	XX/XX/XX	Title: MTS450CA CRICKET RS232	
APRVD X	XX/XX/XX	Size: B	Document Number: 6310-0335-01 Rev: A
		Date: Wednesday, October 20, 2004	Sheet 3 of 6

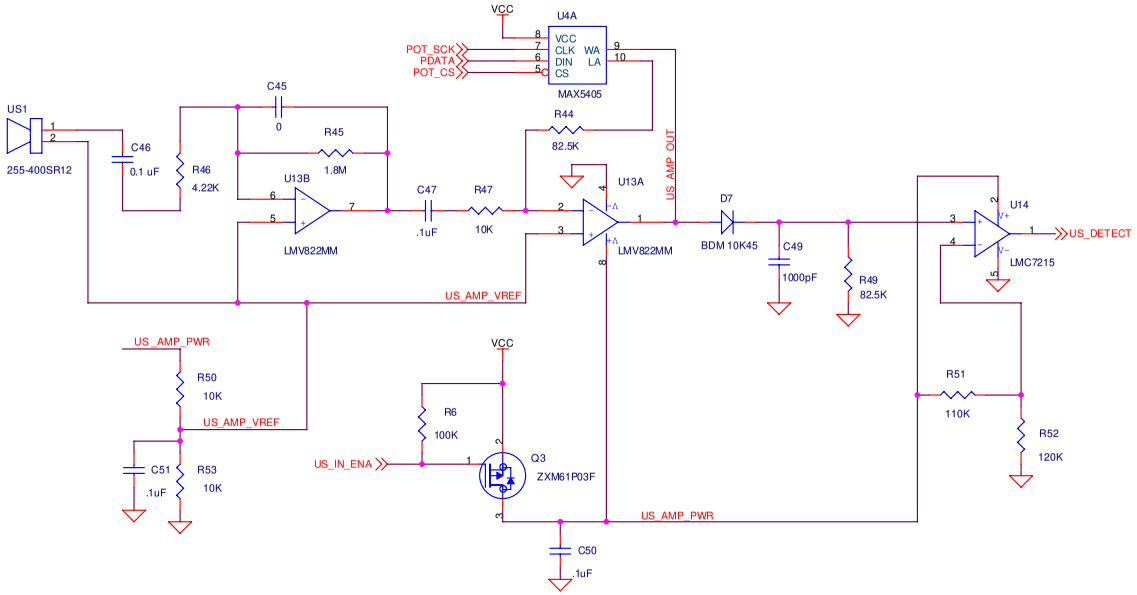






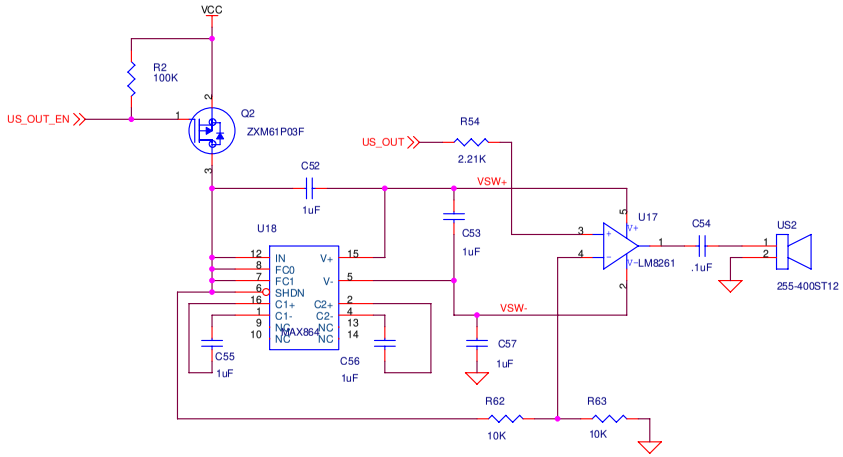
CRICKET V2.0
NETWORKS AND MOBILE SYSTEMS GROUP
CSAIL@MIT
LEAD DESIGNER: NISSANKA B. PRIYANTHA

DWN M. GRIMMER	DATE 9-2-03	Crossbow Technology
CHK M. GRIMMER	9-2-03	41 Daggett Drive San Jose, CA. 95134
APRVD X	XX/XX/XX	Title: MTS450CA CRICKET RS232
APRVD X	XX/XX/XX	Size: B; Document Number: 6310-0335-01; Rev: A
		Date: Wednesday, October 20, 2004; Sheet 4 of 6



CRICKET V2.0
NETWORKS AND MOBILE SYSTEMS GROUP
CSAIL@MIT
LEAD DESIGNER: NISSANKA B. PRIYANTHA

DWN M. GRIMMER	DATE 9-2-03	Crossbow Technology
CHK M. GRIMMER	9-2-03	41 Daggett Drive San Jose, CA. 95134
APRVD X	XX/XX/XX	Title: MTS450CA CRICKET RS232
APRVD X	XX/XX/XX	Size: B; Document Number: 6310-0335-01; Rev: A
		Date: Wednesday, October 20, 2004; Sheet 5 of 6



CRICKET V2.0
NETWORKS AND MOBILE SYSTEMS GROUP
CSAIL@MIT
LEAD DESIGNER: NISSANKA B. PRIYANTHA

DWN M. GRIMMER	DATE 9-2-03	Crossbow Technology 41 Daggett Drive San Jose, CA. 95134
CHK M. GRIMMER	9-2-03	Title: MTS450CA CRICKET RS232
APRVD X	XX/XX/XX	Size: B — Document Number: 6310-0335-01 — Rev: A
APRVD X	XX/XX/XX	Date: Wednesday, October 20, 2004 — Sheet 6 of 6

Appendix B

Design Considerations and Compromises

Here we discuss some of the system design decisions and compromises made when designing the Cricket system.

B.1 Detecting Ultrasonic Signal Arrival

Cricket uses a simple threshold-based approach to detect the arrival of ultrasonic (US) signals. The listener assume the arrival of a US signal when the output of the US amplifiers go beyond 65 mV. As the Section 4.1.3 described, this results in a US received signal strength dependent error in distance estimation. Although a more complicated approach, such as using a sampled US signals to determine and compensate for the signal strength dependent error, is possible, we did not use this approach due to the increased hardware complexity for sampling US signals and the need for more memory to store the samples.

B.2 Modulating the Ultrasonic Signal

As Section 4.2 described, the lack of encoded data on the US signal to determine the source beacon causes distance estimate errors due to beacon transmission interactions. We decided not to encode data on the US signals to keep the US transmitter and receiver components and circuits simple. Although several research groups have successfully transmitted data on the US signals using wide-band US transmitters and receivers, these technologies require complex hardware implementations that can also lead to increased beacon power consumption.

B.3 The Microcontroller

Cricket nodes use the Atmega128L microcontroller. Although it is possible to implement the basic Cricket functionality using a much simpler microcontroller, we decided

to use the Atmega128L because this has enough processing power and storage to use a Cricket node as a sensor platform. On the other hand, we did not use a more powerful microcontroller due to power consumption considerations.

B.4 Host Interface

Cricket nodes use an RS232 interface to attach to host devices. An advantage of RS232 is that most computers have a built-in RS232 interface. However, the slow data rate, bulky connectors, and the need for external wires are some of the disadvantages of the RS232 interface. A USB or a Compact Flash interface can replace the RS232 interface, however these interfaces generally require more complex hardware implementations compared to the RS232 interface.

Appendix C

System Parameters

Parameter	Value	Description
Beacon frequency	1 Hz	Based on the heuristic that a typical application would need only a 1 Hz update rate per beacon.
RF frequency	436 MHz	Need to operate in the ISM band. The 436 MHz band has less interference compared to 900 MHz band.
RF transmit power	-3 dBm	Results in required level of RF coverage in a typical indoor environment.
RF data rate	19.2 kbps	According to the manufacturer data sheet this is an acceptable data rate for the given RF transmit power.
US frequency	40 KHz	Selected based on the availability of transducers and the required coverage. Higher frequency ($\simeq$ 100 kHz) transducers have a narrower coverage angle.
US pulse duration	150 μs	Minimum pulse duration for proper detection at the listener.
US receiver gain	70 to 78 dB	This level of gain provides $\simeq$ 10 m range when listener and beacon are held face to face.
RS232 data rate	115.2 kbps	A compatible data rate for most of the available hosts.

Table C.1: Cricket system parameters.

Bibliography

[1] L. Aalto, N. Göthlin, J. Korhonen, and T. Ojala. Bluetooth and WAP push-based location-aware mobile advertising system. In *MobiSYS '04: Proceedings of the 2nd international conference on mobile systems, applications, and services*, pages 49–58, Boston, MA, 2004. ACM Press.

[2] A. D. Aczel. *The Riddle of the Compass: The Invention that Changed the World*, chapter 7. Harcourt, Orlando, FL, first edition, August 2001.

[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*, pages 186–201, Kiawah Island, SC, December 1999.

[4] Extended AT Command Set. `http://www.modems.com/general/extendat.html`.

[5] Atmega128L datasheet. `http://www.atmel.com/dyn/resources/prod_documents/doc2467.pdf`.

[6] P. Bahl and V. Padmanabhan. RADAR: An In-Building RF-based User Location and Tracking System. In *Proc. IEEE INFOCOM*, pages 775–784, Tel-Aviv, Israel, March 2000.

[7] M. Beigl. Point & click - interaction in smart environments. In *HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, pages 311–313, London, UK, 1999. Springer-Verlag.

[8] B. Berger, J. Kleinberg, and T. Leighton. Reconstructing a three-dimensional model with arbitrary errors. *Journal of the ACM*, 46(2):212–235, 1999.

[9] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Nashua, NH, second edition, June 1999.

[10] R. Bud and D. Warner. *Instruments of Science: An Historical Encyclopedia*, pages 501–503. Garland Encyclopedias in the History of Science. Garland Publishing, New York, NY, first edition, December 1997.

[11] N. Bulusu, J. Heidemann, D. Estrin, and T. Tran. Self-configuring localization systems: Design and experimental evaluation. *ACM Transactions on Embedded Computing Systems*, 3(1):24–60, 2004.

[12] Nirupama Bulusu, John Heidemann, and Deborah Estrin. Gps-less low cost outdoor localization for very small devices. *IEEE Personal Communications Magazine*, 7(5):28–34, October 2000.

[13] Pinhanez C. Creating ubiquitous interactive games using everywhere displays projectors. In *Proc. of the International Workshop on Entertainment Computing (IWEC'02)*, pages 149–156, Makuhari, Japan, May 2002.

[14] A Caruso, S. Chessa, S. De, and A. Urpi1. GPS Free Coordinate Assignment and Routing in Wireless Sensor Networks. In *IEEE INFOCOM*, Miami, FL, March 2005.

[15] P. Castro, P. Chiu, T. Kremenek, and R. Muntz. A probabilistic room location service for wireless networked environments. In *Proc. 3rd UbiComp Conf.*, pages 18–34, Atlanta, GA, 2001.

[16] CC1000 datasheet. `http://www.chipcon.com/files/CC1000_Data_Sheet_2_2.pdf`.

[17] S. Cheshire and M. Baker. Consistent overhead Byte stuffing. *IEEE ACM Transactions on Networking (TON)*, 7(2):159 – 172, April 1999.

[18] W. C. Chung and D. S. Ha. An accurate Ultra Wideband (UWB) Ranging for Precision Asset Location. In *Int. Conf. on Ultra Wideband Systems and Technologies*, pages 389–393, Reston, VA, November 2003.

[19] R. Connelly. On generic global rigidity. In *Applied Geometry and Discrete Mathematics*, volume 4, pages 147–155. American Mathematical Society, 1991.

[20] R. Connelly. Rigidity. In *Handbook of Convex Geometry*, volume A, pages 223–271. North-Holland, Amsterdam, 1993.

[21] P. Corke, R. Peterson, and D. Rus. Networked robots: Flying robot navigation using a sensor net. In *Proc. 11th ISRR*, Siena, Italy, November 2003.

[22] C. Coullard and A. Lubiw. Distance visibility graphs. *International Journal of Computational Geometry and Applications*, 2(4):349–362, 1992.

[23] O. Cramer. The variation of the specific heat ratio and the speed of sound in air with temperature, pressure, humidity, and CO_2 concentration. *Acoustical Society of America Journal*, 93(5):2510–2516, May 1993.

[24] Cricket homepage. `http://cricket.csail.mit.edu`.

[25] G. Crippen and T. Havel. *Distance Geometry and Molecular Conformation*. John Wiley & Sons, Hoboken, NJ, 1988.

[26] Crossbow technology inc. homepage. http://www.xbow.com.

[27] R.S. Davis. Equation for the determination of the density of moist air (1981/91). *Metrologia*, 29(1):67–70, 1992.

[28] M. Dertouzos. The Future of Computing. *Scientific American*, August 1999. Available from http://www.sciam.com/article.cfm?chanID=sa006&colID=1&articleID=000C07%AA-99F2-1C72-9EB7809EC588F2D7.

[29] L. Doherty, K. Pister, and L. Ghaoui. Convex position estimation in wireless sensor networks. In *Proc. IEEE INFOCOM*, pages 1655–1663, Anchorage, AK, April 2001.

[30] Enhanced 911 service. http://www.fcc.gov/911/enhanced/.

[31] H. A. El-Natour, C. Macabiau, M. L. Boucheret, and A. C. Escher. Impact of Multipath and Cross-Correlation on GPS Acquisition in Indoor Environments. In *Institute of Navigation, National Technical Meeting*, San Diego, CA, January 2005.

[32] E. Elnahrawy, X. Li, and R. P. Martin. The Limits of Localization Using Signal Strength: A Comparative Study. In *IEEE Sensor and Ad hoc Communications and Networks Conference (SECON 2004)*, Santa Clara, CA, October 2004.

[33] P. Faltstrom, D. Crocker, and E. Fair. *MIME Content Type for BinHex Encoded Files*, December 1994. RFC 1741 (http://www.ietf.org/rfc/rfc1741.txt).

[34] N. Freed and N. Borenstein. *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*, November 1996. RFC 2045 (http://www.ietf.org/rfc/rfc2045.txt).

[35] Geocaching homepage. http://www.geocaching.com.

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

[37] L. Girod and D. Estrin. Robust range estimation using acoustic and multimodal sensing. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Maui, HI, October 2001.

[38] G. A. Glatzmaier and P. H. Roberts. Rotation and magnetism of Earth's inner core. *Science*, 274(5294):1887–1891, December 1996.

[39] J. Graver, B. Servatius, and H. Servatius. *Combinatorial Rigidity*. American Mathematical Society, Providence, RI, 1993.

[40] J. E. Graver. *Counting on Frameworks: Mathematics to Aid the Design of Rigid Structures*. Mathematical Association of America, Washington, DC, 2001.

[41] W. G. Griswold, R. Boyer, R. S. Brown, T. M. Truong, E. Bhasker, G. R. Jay, and Shapiro R. B. ActiveCampus - Sustaining Educational Communities through Mobile Technology. University of California, San Diego: La Jolla, 2002.

[42] A. Harter and A. Hopper. A New Location Technique for the Active Office. *IEEE Personal Communications*, 4(5):43–47, October 1997.

[43] A. Harter, A. Hopper, P. Steggles, A. Ward, and P. Webster. The Anatomy of a Context-Aware Application. In *Proc. 5th ACM MOBICOM Conf.*, pages 59–68, Seattle, WA, August 1999.

[44] M. Hazas and A. Ward. A novel broadband ultrasonic location system. In *Proceedings of UbiComp 2002: Ubiquitous Computing*, pages 264–280, Göteborg, Sweden, September 2002.

[45] E. Hecht. *Optics*. Addison-Wesley, Reading, MA, fourth edition, 2002.

[46] A. Helfrick. *Principles of Avionics*. Avionics Communications Inc., Leesburg, VA, third edition, 2004.

[47] HMC6352 Honeywell compass module. `http://www.ssec.honeywell.com/magnetic/hmc6352.html`.

[48] B. Hoffmann-Wellenhof, H. Lichtenegger, and J. Collins. *Global Positioning System: Theory and Practice*. Springer-Verlag, New York, NY, fourth edition, 1997.

[49] A. Howard, M. Mataric, and G. Sukhatme. Relaxation on a mesh: A formalism for generalized localization. In *Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 1055–1060, Wailea, HI, October 2001.

[50] G Hulot, C. Eymin, B. Langlais, M. Mandea, and N. Olsen. Small-scale structure of the geodynamo inferred from oersted and magsat satellite data. *Nature*, 416(6881):620–623, April 2002.

[51] IE Logistics - Fleet Management SOlutions Provider. `http://www.ielogistics.net`.

[52] B. Jackson and T. Jordán. Connected rigidity matroids and unique realizations of graphs. Technical Report TR-2002-12, Egerváry Research Group on Combinatorial Optimization, Budapest, Hungary, March 2003.

[53] N. Jones. Anomalies hint at magnetic pole flip. *New Scientist*, April 2002. Available from `http://www.newscientist.com/article.ns?id=dn2152`.

[54] S. Josefsson. *The Base16, Base32, and Base64 Data Encodings*, July 2003. RFC 3548 (`http://www.ietf.org/rfc/rfc3548.txt`).

[55] T. Kindberg, J. Barton, J. Morgan, G. Becker, D. Caswell, P. Debaty, G. Gopal, M. Frid, V. Krishnan, H. Morris, J. Schettino, B. Serra, and M. Spasojevic. People, Places, Things: Web Presence for the Real World. In *Proceedings of WMCS 2000*, pages 365–376, Monterey,CA, December 2000.

[56] A. M. Ladd, K. E. Bekris, A. Rudys, G. Marceau, L. E. Kavraki, and D. S. Wallach. Robotics-based location sensing using wireless Ethernet. In *Proc. 8th ACM MOBICOM Conf.*, pages 227–238, Atlanta, GA, September 2002.

[57] B. P. Lathi. *Modern Digital and Analog Communications Systems*. Oxford University Press, New York, NY, third edition, 1998.

[58] David R. Lide. *CRC Handbook of Chemistry and Physics*. CRC Press, Boca Raton, FL, 85th edition, 2005.

[59] LoJack Car Security System for Stolen Vehicle Recovery. `http://www.privacyinternational.org/survey/technologies.html`.

[60] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. TAG: a Tiny AGregation Service for Ad-Hoc Sensor Networks. In *Proceedings of the Fifth USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 131–146, Boston, MA, December 2002.

[61] A. K. L. Miu. Design and Implementation of an Indoor Mobile Navigation System. Master's thesis, Massachusetts Institute of Technology, May 2002.

[62] M. Montemerlo, J. Pineau, N. Roy, S. Thrun, and V. Verma. Experiences with a mobile robotic guide for the elderly. In *Eighteenth national conference on Artificial intelligence*, pages 587–592, Edmonton, Alberta, Canada, 2002. American Association for Artificial Intelligence.

[63] D. Moore, J. Leonard, D. Rus, and S. S. Teller. Robust distributed network localization with noisy range measurements. In *Proceedings of SenSys 2004*, pages 50–61, Baltimore, MD, November 2004.

[64] Moses, R.L. and Krishnamurthy, D. and Patterson, R. M. A self-localization method for wireless sensor networks. *Eurasip Journal on Applied Signal Processing, Special Issue on Sensor Networks*, pages 348–358, 2001.

[65] MSP430 microcontroller. `http://www.ti.com/msp430`.

[66] R. Nagpal, H. Shrobe, and J. Bachrach. Organizing a Global Coordinate System from Local Information on an Ad Hoc Sensor Network. In *Proc. International Workshop on Information Processing in Sensor Networks, IPSN 2003*, pages 333–348, Palo Alto, CA, 2003.

[67] D. Niculescu and B. Nath. Ad Hoc Positioning System (APS) Using AOA. In *Proc. of IEEE INFOCOM*, pages 2037–2040, Salt Lake City, UT, April 2003.

[68] J. R. Nogueras. A Stream Redirection Architecture for Pervasive Computing Environments. Master's thesis, Massachusetts Institute of Technology, May 2001.

[69] P. Pathirana, A. Savkin, S. Jha, and N. Bulusu. Node localization using mobile robots in delay-tolerant sensor networks. *IEEE Trans. Mobile Computing*, 4(3):285–296, May 2005.

[70] N. Priyantha, H. Balakrishnan, E. Demaine, and S. Teller. Anchor-Free Distributed Localization in Sensor Networks. Technical Report 892, MIT Laboratory for Computer Science, April 2003.

[71] N. Priyantha, A. Chakraborty, and H. Balakrishnan. The Cricket Location-Support System. In *Proc. 6th ACM MOBICOM Conf.*, pages 32–43, Boston, MA, August 2000.

[72] N. Priyantha, A. Miu, H. Balakrishnan, and S. Teller. The Cricket Compass for Context-Aware Mobile Applications. In *Proc. 7th ACM MOBICOM Conf.*, pages 1–14, Rome, Italy, July 2001.

[73] N. B. Priyantha, H. Balakrishnan, E. Demaine, and S. Teller. Mobile-Assisted Localization in Wireless Sensor Networks. In *IEEE INFOCOM*, Miami, FL, March 2005.

[74] K. K. Ramakrishnan. Scheduling issues for interfacing to high speed networks. In *Proc. Globecom 1992 IEEE Global Telecommunications Conf.*, pages 622–626, Orlando, FL, 1992.

[75] A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, and I. Stoica. Geographic routing without location information. In *Proc. 9th ACM MOBICOM*, pages 96–108, San Diego, CA, September 2003.

[76] T. S. Rappaport. *Wireless Communications: Principles and Practice*. Prentice Hall, Upper Saddle River, NJ, second edition, 2001.

[77] J. Romkey. *A Nonstandard For Transmission Of IP Datagrams Over Serial Lines: SLIP*, June 1988. RFC 1055 (`http://www.ietf.org/rfc/rfc1055.txt`).

[78] Merrill I. S. *Introduction to Radar Systems*. McGraw-Hill, New York, NY, third edition, December 2002.

[79] S. Saha, K. Chaudhuri, D. Sanghi, and P. Bhagwat. Location determination of a mobile device using IEEE 802.11 access point signals. In *IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1987–1992, New Orleans, LA, March 2003.

[80] J. S. Sandhu, A. M. Agogino, and A. K. Agogino. Wireless sensor networks for commercial lighting control: Decision making with multi-agent systems. In *Proceedings of the AAAI Workshop on Sensor Networks*, San Jose, CA, July 2004.

[81] C. Savarese, J. Rabaey, and J. Beutel. Locationing in Distributed Ad-Hoc Wireless Sensor Networks. In *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 2037–2040, Salt Lake City, UT, May 2001.

[82] C. Savarese, J. Rabaey, and K. Langendoen. Robust Positioning Algorithms for Distributed Ad-Hoc Wireless Sensor Networks. In *USENIX Annual Technical Conference*, pages 317–327, Monterey, CA, June 2002.

[83] A. Savvides, W. Garber, A. Sachin, R. Moses, and M. Srivastava. On the Error Characteristics of Multihop Node Localization in Ad-Hoc Sensor Networks. In *Proc. International Workshop on Information Processing in Sensor Networks*, pages 317–3–32, Palo Alto, CA, 2003.

[84] A. Savvides, C. Han, and M. Srivastava. Dynamic Fine-Grained Localization in Ad-Hoc Networks of Sensors. In *Proc. 7th ACM MOBICOM Conf.*, pages 166–179, Rome, Italy, July 2001.

[85] A. Savvides, H. Park, and M. Srivastava. The Bits and Flops of the N-hop Multilateration Primitive For Node Localization Problems. In *Proceedings of the First ACM International Workshop on Wireless Sensor Networks and Applications*, pages 112–121, Atlanta, GA, September 2002.

[86] J. B. Saxe. Embeddability of weighted graphs in k-space is strongly NP-hard. In *Proceedings of the 17th Allerton Conference on Communications, Control, and Computing*, pages 480–489, Urbana, IL, 1979.

[87] J. Scott and M. Hazas. User-Friendly Surveying Techniques for Location-Aware Systems. In *Proc. 5th UbiComp*, pages 44–53, Seattle, WA, October 2003. Springer-Verlag.

[88] Y. Shang, W. Ruml, Y. Zhang, and M. Fromherz. Localization from mere connectivity. In *Proc. of the fourth ACM international symposium on mobile ad hoc networking and computing, (MOBIHOC)*, pages 201–212, Annapolis, MD, 2003.

[89] E. Shih, P. Bahl, and M.J. Sinclair. Wake on wireless: An event driven energy saving strategy for battery operated devices. In *Proc. 8th ACM MOBICOM Conf.*, pages 160–171, Atlanta, GA, USA, 2002.

[90] M. Sichitiu and V. Ramadurai. Localization of wireless sensor networks with a mobile beacon. In *Proc. 1st IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, pages 174–183, Fort Lauderdale, FL, October 2004.

[91] W. Simpson. *PPP in HDLC-like Framing*, July 1994. RFC 1662 (`http://www.ietf.org/rfc/rfc1662.txt`).

[92] A. Smith, H. Balakrishnan, M. Goraczko, and N. Priyantha. Tracking Moving Devices with the Cricket Location System. In *2nd International Conference on Mobile Systems, Applications and Services (Mobisys)*, pages 190–202, Boston, MA, June 2004.

[93] X. Song and P. Richards. Seismological evidence for differential rotation of the earth's inner core. *Nature*, 382(6588):221–224, July 1996.

[94] NPL Acoustics: Calculation of speed of sound in air. `http://www.npl.co.uk/acoustics/techguides/speedair/`.

[95] M.A. Spirito. Accuracy of Hyperbolic Mobile Station Location in Cellular Networks. *Electronics Letters*, 37(11):708–710, 2001.

[96] M. Srivastava, R. Muntz, and M. Potkonjak. Smart kindergarten: sensor-based wireless networks for smart developmental problem-solving enviroments. In *Proc. 7th ACM MOBICOM Conf.*, pages 132–138, Rome, Italy, 2001.

[97] N. Sukaviriya, M. Podlaseck, R. Kjeldsen, A. Levas, G. Pingali, and C. Pinhanez. Embedding interactions in a retail store environment: The design and lessons learned. In *Proc. of the Ninth IFIP International Conference on Human-Computer Interaction*, Zurich, Switzerland, 2003.

[98] Gita Sukthankar. The dynadoom visualization agent: A handheld interface for live action gaming. In *Workshop on Ubiquitous Agents on Embedded, Wearable, and Mobile Devices (Conference on Intelligent Agents and Multiagent Systems)*, Bologna, Italy, July 2002.

[99] S. Teller, J. Chen, and H. Balakrishnan. Pervasive pose-aware applications and infrastructure. *IEEE Computer Graphics and Applications*, 23(4):14–18, 2003.

[100] Tinyos operating system. `http://www.tinyos.net`.

[101] Ubisense homepage. `http://www.ubisense.net`.

[102] 255-400ST16 and 255-400SR16 ultrasonic transducer datasheet. `http://www.mouser.com/catalog/specsheets/062603.pdf`.

[103] U.S. Environmental Protection Agency. *Building Air Quality: A Guide for Building Owners and Facility Managers*, chapter Appendix C, pages 141–146. National Center for Environmental Publications (NSCEP), Cincinnati, OH.

[104] Ultra Wideband working group homepage. `http://www.uwb.org`.

[105] 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.

[106] WayFinder Systems homepage. `http://www.wayfinder.com`.

[107] G. Welch and G. Bishop. SCAAT: Incremental tracking with incomplete information. *Computer Graphics*, 31(Annual Conference Series):333–344, 1997.

[108] Andrew Wilson and Steven Shafer. XWand: UI for intelligent spaces. In *Proc. of the SIGCHI conference on Human factors in computing systems*, pages 545–552, New York, NY, 2003. ACM Press.

[109] M. Youssef, A. Agrawala, and A. U. Shankar. WLAN Location Determination via Clustering and Probability Distributions. In *IEEE Pervasive Computing and Communications Conference(PerCom)*, page 143, Fort Worth, TX, March 2003.