1 Path exposure, target location, classification and tracking in sensor networks

Kousha Moaveni-Nejad, Xiang-Yang Li

Department of Computer Science, Illinois Institute of Technology

10 West 31st Street, Chicago, IL 60616.

Email: moavkoo@iit.edu, xli@cs.iit.edu.

Contents

1.1 INTRODUCTION

Wireless sensor networks are large-scale distributed embedded systems composed of small devices that integrate sensors, actuators, wireless communication, and microprocessors. With advances in hardware, it will soon be feasible to deploy dense collections of sensors to perform distributed micro-sensing of physical environments. Sensor networks will serve as a key infrastructure for a broad range of applications including precision agriculture, intelligent highway systems, emergent disaster recovery, and surveillance.

Sensor networks are an emerging technology that promises unprecedented ability to monitor and instrument the physical world. Sensor networks consist of a large number of inexpensive wireless devices (nodes) densely distributed over the region of interest and have wireless connectivity. They are typically battery powered with limited communication and computation abilities. Also each node in a wireless sensor network is equipped with a variety of sensing modalities, such as acoustic, seismic, and infrared.

Having location information can be very useful and has so many applications, it can answer questions like: Are we almost to the campsite? What lab bench was I standing by when I prepared these tissue samples? How should our search-and-rescue team move to quickly locate all the avalanche victims? Can I automatically display this stock devaluation chart on the large screen I am standing next to? Where is the nearest cardiac defibrillation unit? and so on.

Service providers can also use location information to provide some novel location-aware services. The navigation system based on GPS is an example. A user can tell the system his destination and the system will guide him there. Phone systems in an enterprise can exploit locations of people to provide follow-me services.

Researchers are working to meet these and similar needs by developing systems and technologies that automatically locate people, equipment, and other tangibles. Indeed, many systems over the years have addressed the problem of automatic location-sensing. Because each approach solves a slightly different problem or supports different applications, they vary in many parameters,such as the physical phenomena used for location determination, the form factor of the sensing apparatus, power requirements, infrastructure versus portable elements, and resolution in time and space.

For outdoor environments, the most well-known positioning system is the global positioning system (GPS) [28], which uses 24 satellites set up by the U.S. Department of Defense to enable global three dimensional positioning services and it provides the accuracy around 20 to 50 m. In addition to the GPS system, positioning can also be done using some wireless networking infrastructures. Taking the PCS cellular networks as an example, the $E911$ emergency service requires determining the location of a phone call via the base stations of the cellular system.

In Global Positioning System (GPS), triangulation uses ranges to at least four known satellites to find the coordinates of the receiver, and the clock bias of the re-

ceiver. For our node location purposes, we are using a simplified version of the GPS triangulation, as we only deal with distances, and there is no need for clock synchronization. Because of the following reasons GPS is not suitable for wireless sensor networks and much work has been dedicated recently to positioning and location tracking in the area of wireless sensor networks. The reasons are as follows:

- It is not available in an indoor environment because satellite signals cannot penetrate buildings.
- For more fine-grained applications, higher accuracy is usually necessary in the positioning result.
- Sensor networks have their own battery constraint, which requires special design.

Many applications of sensor networks require knowledge of physical sensor positions. For example, target detection and tracking is usually associated with location information [23]. Further, knowledge of sensor location can be used to facilitate network functions such as packet routing [9],[18], [24], and collaborative signal processing [14]. Sensor position can also serve as a unique node identifier, making it unnecessary for each sensor to have a unique ID assigned prior to its deployment.

In sensor networks the capabilities of individual nodes are very limited and nodes are often powered by batteries only. To conserve energy collaboration between nodes is required and communication between nodes should be minimized. To achieve this goal nodes in wireless sensor networks (WSNs) need to determine devices context and since each node has limited power, we want to determine the location of individual sensor nodes without relying on external infrastructure (base stations, satellites, etc.).

Location information not only can be used to minimize the communication but also can be used to improve the performance of wireless networks and provide new types of services. For example, it can facilitate routing in a wireless ad hoc network to reduce routing overhead. This is known as geographic routing [20, 25]. Through location-aware network protocols, the number of control packets can be reduced. Other types of location-based services include geocast [19], by which a user can request to send a message to a specific area, and temporal geocast, by which a user can request to send a message to a specific area at specific time. In contrast to traditional multicast, such messages are not targeted at a fixed group of members, but rather at members located in a specific physical area.

However, location discovery in wireless sensor networks is very challenging. First, the positioning algorithm must be distributed and localized in order to scale well for large sensor networks. Second, the localization protocol must minimize communication and computation overhead for each sensor since nodes have very limited resources (power, CPU, memory, etc.). Third, the positioning functionality should not increase the cost and complexity of the sensor since an application may require thousands of sensors. Fourth, a location detection scheme should be robust.

It should work with accuracy and precision in various environments, and should not depend on sensor to sensor connectivity in the network.

The localization problem has received considerable attention in the past, as many applications need to know where objects or persons are, and hence various location services have been created. Undoubtedly, the Global Positioning System (GPS) is the most well-known location service in use today. The approach taken by GPS, however, is unsuitable for low-cost, ad-hoc sensor networks since GPS is based on extensive infrastructure (i.e., satellites). Likewise solutions developed in the area of robotic [6, 22, 36] and ubiquitous computing [16] are generally not applicable for sensor networks as they require too much processing power and energy. Recently a number of localization systems have been proposed specifically for sensor networks [10, 11, 26]. We are interested in truly distributed algorithms that can be employed on large-scale ad-hoc sensor networks (100+ nodes). Such algorithms should be:

- self-organizing (i.e., do not depend on global infrastructure).
- robust (i.e., be tolerant to node failures and range errors).
- energy efficient (i.e., require little computation and, especially, communication).

These requirements immediately rule out some of the proposed localization algorithms for sensor networks. The rest of this chapter is organized as follows. In Section 1.2, we briefly discuss what informations are available to the nodes whose location is unknown and then the methods that these information can be used to derive the location of the object are discussed. In Section 1.3 the localization problem is introduced and methods to solve it are discussed. Similarly in Section 1.4 the target tracking problem is introduced and methods to solve it are discussed. Section 1.5 reviews some experimental location and tracking system and finally section 1.6 concludes this book chapter.

1.2 NAVIGATION TECHNIQUES TO DERIVE LOCATION

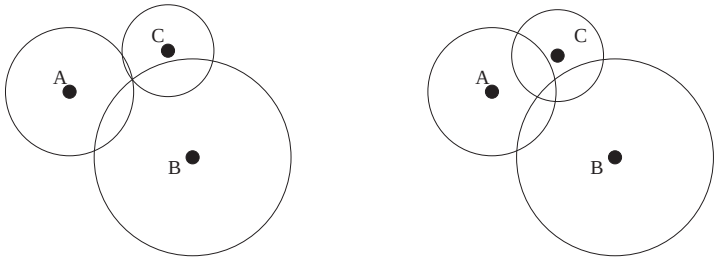
In previous chapter we discussed several approaches with which an object can estimate its distance or its relative location to a reference point. In this section several methods used to calculate an object's location will be discussed.

1.2.1 Lateration

Lateration computes the position of an object by measuring its distance from multiple reference positions. Calculating an objects position in two dimensions requires distance measurements from 3 non-collinear points as shown in Figure 1.1(a). In 3 dimensions, distance measurements from 4 non-coplanar points are required. Domain-specific knowledge may reduce the number of required distance measurements. For

example, the Active Bat Location System measures distance from indoor mobile tags, called Bats, to a grid of ceiling mounted ultrasound sensors [13]. A Bat's 3-dimensional position can be determined using only 3 distance measurements because the sensors in the ceiling are always above the receiver. The geometric ambiguity of only 3 distance measurements can be resolved because the Bat is known to be below the sensors and not in the alternate possible position on the next floor or roof above the sensor grid. Following two different lateration technics are described.

1.2.1.1 Trilateration Trilateration is a well-known technique in which the positioning system has a number of $beacons$ at known locations. These beacons can transmit signals so that other devices can determine their distances to these beacons based on received signals. If a device can hear at least three beacons, its location can be estimated. Figure 1.1(a) shows how trilateration works: A, B, and C are beacons with known locations. From A's signal, one can determine the distance to A and thus that the object should be located at the circle centered at A with radius equal to estimated distance. Similarly, from B's and C's signals, it can be determined that the object should be located at some circles centered at B and C, respectively. Thus, the intersection of the three circles is the estimated location of the device. The preceding discussion has assumed an ideal situation; however, as mentioned earlier, distance estimation always contains errors that will, in turn, lead to location errors. Figure 1.1(b) illustrates an example in practice. The three circles do not intersect in a common point. In this case, the maximum likelihood method may be used to estimate the devices location. Let the three beacons A, B, and C be located at (x_A, y_A),(x_B, y_B), and (x_C, y_C), respectively. For any point (x, y) on the plane, a difference function is computed:



(a) Ideal Situation (b) Real situation with error.

Fig. 1.1 Trilateration Method.

$$\sigma_{x,y} = |\sqrt{(x-x_A)^2+(y-y_A)^2} - r_A| + |\sqrt{(x-x_B)^2+(y-y_B)^2} - r_B| + |\sqrt{(x-x_C)^2+(y-y_C)^2} - r_C|$$

Where r_A, r_B, and r_C are the estimated distances to A, B, and C, respectively. The location of the object can then be predicted as the point (x, y) among all points

such that $\sigma_{x,y}$ is minimized. In addition to using the ToA approach for positioning, the AoA approach can be used. For example, in Figure 1.2, the unknown node D measures the angle of, $\angle ADB$, $\angle BDC$, and $\angle ADC$ by the received signals from beacons A, B, and C. From this information, $D^{'}$s location can be derived [26].

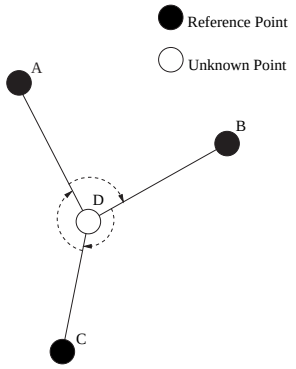


Fig. 1.2 Angle measurement from three beacons A, B, and C

1.2.1.2 Multilateration The trilateration method has its limitation in that at least three beacons are needed to determine a devices location. In a sensor network, in which nodes are randomly deployed, this may not be true. Several multilateration methods are proposed to relieve this limitation. The AHLoS (Ad Hoc Localization System) [34] is a distributed system for location discovery. In the network, some beacons have known locations and some devices have unknown locations. The AHLoS enables nodes to discover their locations by using a set of distributed iterative algorithms. The basic one is atomic multilateration, which can estimate the location of a device of unknown location if at least three beacons are within its sensing range. Figure 1.3 shows an example in which, initially, beacon nodes contain only nodes marked as reference point. Device nodes A, B, and C are at unknown locations. In the first iteration, as Figure 1.3(a) shows, the locations of nodes A and C will be determined. Note that since node B can communicate with only one beacon, it's location cannot be determined.

The atomic multilateration is further extended to an iterative multilateration method. Specifically, once the location of a device is estimated, its role is changed to a beacon node so as to help determining other devices locations. This is repeated until all hosts' locations are determined (if possible). As Figure 1.3(b) shows, in the second iteration, the location of node B can be determined with the help of nodes A and B, which are now serving as beacons. The iterative multilateration still has its limitation. For example, as Figure 1.4 shows, it is impossible to determine node $2'$s and node $4'$s locations even if the locations of nodes 1, 3, 5, and 6 are known. The collaborative multilateration method may relieve this problem because it allows one to predict multiple potential locations of a node if it can hear fewer than three beacons. For example, in Figure 1.4, from beacon nodes 1 and 3, two potential locations of

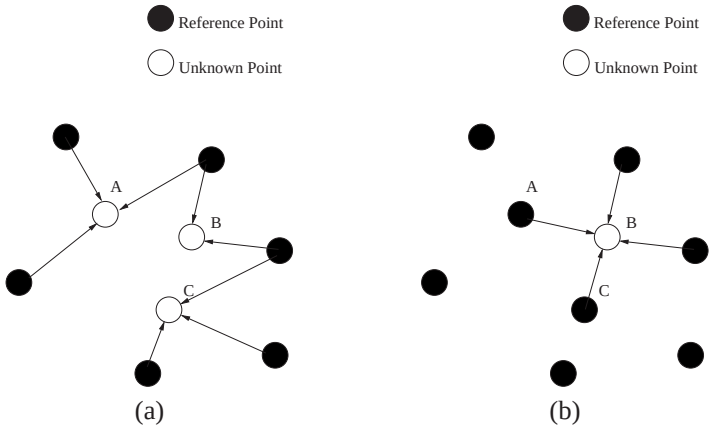


Fig. 1.3 (a) Atomic multilateration; (b)Iterative multilateration

node 2 may be guessed (the other potential location is marked by $2'$). Similarly, from beacon nodes 5 and 6, one may guess two potential locations of node 4 (the other potential location is marked by $4'$). Collaborative multilateration allows estimation of the distance between nodes 2 and 4. With this information, the locations of nodes 2 and 4 can be estimated, as the figure shows.

Note that, in a distributed location protocol, we may wonder if the order of determining the position of nodes can affect the outcome of the location system. If the distance or regular estimation is precise, we can show that the order in which each node determines its location and then serves as beacon node will *not* affect the number of nodes whose positions can be computed. However, when the information is not precise, it does affect this and further, the precision of the system.

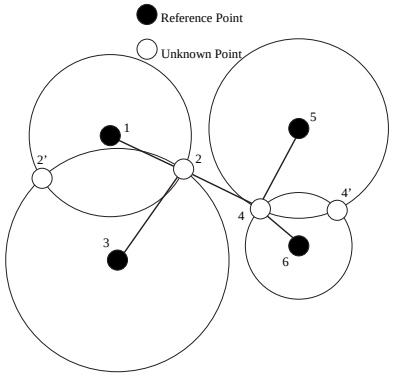


Fig. 1.4 Collaborative multilateration in AHLoS

1.2.2 Pattern Matching Using Database

Another type of location discovery is by pattern matching. Instead of estimating the distance between a beacon and a device, this approach tries to compare the received signal pattern against the training patterns in the database. Thus, this method is also known as the *fingerprinting* approach. The basic idea is that signal strength received at a fixed location is not necessarily a constant. It typically moves up and down, so it would be better to model signal strength by a random variable. This is especially true for indoor environments.

The main idea is to compare the received signals against those in the database and determine the likelihood that the device is currently located in a position. A typical solution has two phases :

- **Off-line phase:** The purpose of this phase is to collect signals from all base stations at each training location. The number of training locations is decided first. Then, the received signal strengths are recorded (for a base station that is too far away, the signal strength is indicated as zero). Each entry in the database has the format: $(x, y, \langle ss_1, ss_2, ..., ss_n \rangle)$, where (x, y) is the coordinate of the training location, and ss_i $(i = 1...n)$ is the signal strength received at the training location from the i^{th} base station. These entries are stored in the database. Note that for higher accuracy, one may establish multiple entries in the database for the same training location. From the database, some positioning rules, which form the positioning model, will then be established.
- **Real-time phase:** With a well-trained positioning model, one can estimate a device's location given the signal strengths collected by the device from all possible base stations. The positioning model may determine a number of locations, each associated with a probability. However, the typical solution is to output only the location with the highest likelihood.

There are several similarity searching methods in the matching process; two approaches are discussed next.

1.2.2.1 Nearest Neighbor Algorithms The simplest approach is the *nearest neighbor in signal space* (NNSS) approach [7, 8]. In the first phase, only the average signal strength of each base station at each training location is recorded. Then, in the second phase, the NNSS algorithm computes the Euclidean distance in signal space between the received signal and each record in the database. Euclidean distance means the square root of the summation of square of the difference between each received signal strength and the corresponding average signal strength from the access point under consideration. The training location with the minimum Euclidean distance is then chosen as the estimated location of the device. Because this algorithm only picks existing locations in the database, to improve its accuracy, it is suggested that the training set be dense enough. One variant of the basic NNSS algorithm is NNSS-AVG. To take the uncertainty of a device's location into consideration, this method tries to pick a small number of training locations that closely

match the received signal strengths (such as those with smaller Euclidean distances). Then, it infers the location of the device to be a function of the coordinates of the selected training locations. For example, one may take the average of the x and y coordinates of the selected training locations as the estimated result.

1.2.2.2 Probability-Based Algorithms The probability-based positioning approach regards signal strength as a probability distribution [30]. In NNSS, because the received signal strengths are averaged out, the probability distribution would disappear. So the probability-based approach will try to maintain more complete information of signal strength distribution. The prediction result is typically more accurate. The core of the probability-based model is the Bayes rule:

$$p(l|o) = \frac{p(o|l)p(l)}{p(o)} = \frac{p(o|l)p(l)}{\sum_{l' \in L} p(o|l')p(l')}$$

Where $p(l|o)$ is the probability that the device is at location l given an observed signal strength pattern o. The prior probability that a device is resident at l is $p(l)$, which may be inferred from history or experience. For example, people may have a higher probability to appear in a hallway or lobby. If this is not available, $p(l)$ may be assumed to be a uniform distribution. L is the set of all training locations. The denominator $p(o)$ does not depend on the location variable l, so it can be treated as a normalized constant whenever only relative probabilities are required. The term $p(o|l)$ is called the likelihood function; this represents the core of the positioning model and can be computed in the off-line phase.

There are two ways to implement the likelihood function [30]:

- **Kernel method:** For each observation o_i in the training data, it is assumed that the signal strength exhibits a Gaussian distribution with mean o_i and standard deviation σ, where σ is an adjustable parameter in the model. Specifically, given o_i, the probability to observe o is

  $$K(o; o_i) = \frac{1}{\sqrt{2\pi}\sigma} exp(\frac{(o - o_i)^2}{2\sigma^2})$$

  Based on the kernel function, the probability $p(o|l)$ can be defined as

  $$p(o|l) = \frac{1}{n_1} \sum_{l_i \in L, i=l} K(o; o_i)$$

  Where n_1 is the number of training vectors in L obtained at location l. Intuitively, the probability function is a mixture of n_1 equally weighted density functions. Also note that the preceding formulas are derived assuming that only one base station exists. With multiple base stations, the probability function will be multivariated, and the probability will become the multiplication of multiple independent probabilities, each for one base station.

- **Histogram method:** Another method to estimate the density functions is to use histogram, which is related to discretization of continuous values to discrete ones. A number of bins can be defined as a set of non-overlapping intervals that cover the whole random variables. The advantage of this method is in its ease in implementation and low computational cost. Another reason is that its discrete property can smooth out the instability of signal strengths.

The probability-based methods can adapt to different environments. To further reduce the computational overhead, Youssef and colleagues [39] proposed a method by clustering training data in the database.

1.2.3 Network Based Tracking

Special concerns such as power saving, bandwidth conservation, and fault tolerance arise when a solution is designed for a wireless sensor network. At the network level, location tracking may be done via the cooperation of sensors. Tseng and colleagues [37] addressed these issues using an agent-based paradigm. Once a new object is detected by the network, a mobile agent will be initiated to track the roaming path of the object. The agent is mobile because it will choose the sensor closest to the object to stay. The agent may invite some nearby slave sensors to cooperatively position the object and inhibit other irrelevant (i.e., farther) sensors from tracking the object. More precisely, only three agents will be used for the tracking purpose at any time and they will move as the object moves. The trilateration method is used for positioning. As a result, the communication and sensing overheads are greatly reduced. Because data transmission may consume a lot of energy, this agent-based approach tries to merge the positioning results locally before sending them to the data center. These authors also address how to conduct data fusion.

Figure 1.5 shows an example. The sensor network is deployed in a regular manner and it is assumed that each sensors sensing distance equals the distance between two neighboring sensors. Initially, each sensor is in the idle state, searching for new objects. Once detecting a target, a sensor will transit to the election state, trying to serve as the master agent. The nearest sensor will win. The master agent will then dispatch two neighboring sensors as the slave agents; master and slave agents will cooperate to position the object. In the figure, the object is first tracked by sensors $\{S_0, S_1, S_2\}$ when resident in A_0, then by $\{S_0, S_2, S_3\}$ when in A_1, by $\{S_2, S_3, S_5\}$ when in A_2, etc. The master agent is responsible for collecting all sensing data and performing the trilateration algorithm. It also conducts data fusion by keeping the tracking results while it moves around. At a proper time, the master agent will forward the tracking result to the data center. Two strategies are proposed for this purpose: threshold-based (TB) strategy, which will forward the result when the amount of data reaches a predefined threshold value T, and distance-based (DB) strategy, which will make a decision based on the routing distance from the agents current location to the data center and the direction in which the agent is moving.

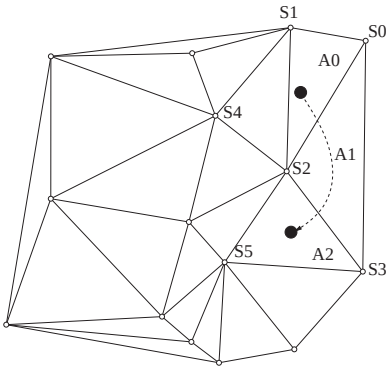


Fig. 1.5 Roaming path of an object (dashed line).

1.3 LOCALIZATION IN WIRELESS SENSOR NETWORKS

In this section we address the issue of localization in ad-hoc sensor networks. That is, we want to determine the location of individual sensor nodes without relying on external infrastructure (base stations, satellites, etc.). Undoubtedly, the Global Positioning System (GPS) is the most well-known location service in use today. The approach taken by GPS, however, is unsuitable for low-cost, ad-hoc sensor networks since GPS is based on extensive infrastructure (i.e., satellites).

Location service enables routing in sufficiently isotropic large networks, without the use of large routing tables. APS (Ah hoc Positioning System)[26] is a distributed, hop by hop positioning algorithm, that works as an extension of both distance vector routing and GPS positioning in order to provide approximate location for all nodes in a network where only a limited fraction of nodes have self location capability. Also, APS is appropriate for indoor location aware applications, when the network's main feature is not the unpredictable, highly mobile topology, but rather deployment that is temporary, and ad hoc. Large sensor networks of low power nodes face a number of challenges: routing without the use of large conventional routing tables, adaptability in front of intermittent functioning regime, network partitioning and survivability. In this section, we address the problem of self locating the nodes in the field, which may provide a solution to the first challenge, and solve other practical problems as well.

Before discussing distributed localization in detail, we first outline the context in which these algorithms have to operate. A first consideration is that the requirement for sensor networks to be self-organizing implies that there is no fine control over the placement of the sensor nodes when the network is installed (e.g., when nodes are dropped from an airplane). Consequently, we assume that nodes are randomly distributed across the environment. For simplicity and ease of presentation we limit the environment to 2 dimensions, but all algorithms are capable of operating in 3D. Figure 1.6 shows an example network with 25 nodes; pairs of nodes that can com-

municate directly are connected by an edge. The connectivity of the nodes in the network (i.e., the average number of neighbors) is an important parameter that has a strong impact on the accuracy of most localization algorithms. It can be set initially by selecting a specific node density, and in some cases it can be set dynamically by adjusting the transmit power of the RF radio in each node.

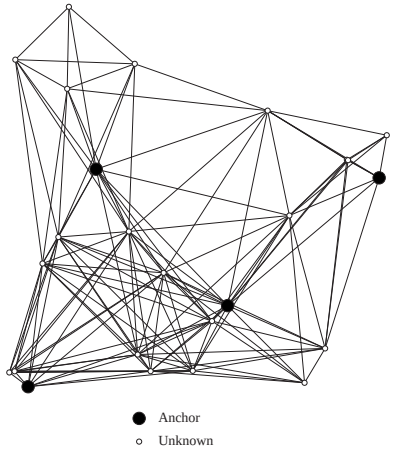


Fig. 1.6 Example network topology

In some application scenarios, nodes may be mobile. Here, however, we focus on static networks, where nodes do not move, since this is already a challenging condition for distributed localization. We assume that some anchor nodes have a priori knowledge of their own position with respect to some global coordinate system. Note that anchor nodes have the same capabilities (processing, communication, energy consumption, etc.) as all other sensor nodes with unknown positions. Ideally the fraction of anchor nodes should be as low as possible to minimize the installation costs. The final element that defines the context of distributed localization is the capability to measure the distance between directly connected nodes in the network. From a cost perspective it is attractive to use the RF radio for measuring the range between nodes, for example, by observing the signal strength. As mentioned before, this approach yields poor distance estimates. Much better results are obtained by time-of-flight measurements (i.e., TOA and TDOA), particularly when acoustic and RF signals are combined; accuracies of a few percent of the transmission range are reported. It is important to realize that the main three context parameters (connectivity, anchor fraction, and range errors) are dependent. Poor range measurements can be compensated for by using many anchors and/or a high connectivity.

1.3.1 Ad-hoc Positioning System (APS)

1.3.1.1 Platform If a graph is *sufficiently* connected, and the lengths of its edges are all known, then its plane topology may be reconstructed [26]. But what

is a sufficient degree of connectivity? If we assimilate the graph with a wire frame, where nodes act as hinges, the goal is to determine which part of the graph has non-coordinate moving parts, and those will be the nodes which can determine their location. Once such a wire-frame is fixed, it will have a reference system of its own, that eventually has to be aligned to the global coordinate system of the GPS. In order to fix this wire frame somewhere on the global plane, at least three nodes (called landmarks), that are GPS enhanced, or know their position by some other means, have to be present in the connected graph. Devices as simple as the Rene motes [17] have software access to the signal strength of the radio signal, thus offering a way to estimate distance to immediate neighbors. This measurements however, are affected by errors. One of the aims of APS is to enhance position accuracy as the fraction of landmarks of the entire population increases. Even if it is theoretically sufficient to have three landmarks, the presence of measurement errors will demand higher fractions of landmarks, depending on the requirements of the application.

1.3.1.2 Algorithm It is not desirable to have the landmarks emit with large power to cover the entire network for several reasons: collisions in local communication, high power usage, coverage problems when moving. Also, it is not acceptable to assume some fixed positions for the landmarks, as the applications envisioned by APS systems are either in flight deployments over inaccessible areas, or possibly involving movement and reconfiguration of the network. In this case, one option is to use hop by hop propagation capability of the network to forward distances to landmarks. In general, they aim for the same principle as GPS, with the difference that the landmarks are contacted in a hop by hop fashion, rather than directly, as ephemerides are. Once an arbitrary node has estimates to a number (≥ 3) of landmarks, it can compute its own position in the plane, using a similar procedure with the one used in GPS position calculation described in the previous section. The estimate we start with is the centroid of the landmarks collected by a node. In what follows we will refer to one landmark only, as the algorithm behaves identically and independently for all the landmarks in the network. It is clear that the immediate neighbors of the landmark can estimate the distance to the landmark by direct signal strength measurement. Using some propagation method, the second hop neighbors then are able to infer their distance to the landmark, and the rest of the network follows, in a controlled flood manner, initiated at the landmark. Complexity of signaling is therefore driven by the total number of landmarks, and by the average degree of each node. What makes this method similar with the distance vector routing, is that at any time, each node only communicates with its immediate neighbors, and in each message exchange it broadcasts available estimates to landmarks acquired so far. This is appropriate for nodes with limited capabilities, which do not need, and cannot handle the image of the entire, possibly moving, network. The APS system used three methods of hop to hop distance propagation and examine advantages and drawbacks for each of them. Each propagation method is appropriate for a certain class of problems as it influences the amount of signaling, power consumption, and position accuracy achieved.

1.3.1.3 Distance to Anchors Nodes that can communicate with anchor nodes directly are able to find their distance to anchor nodes , but this information is not available to all nodes. Nodes share information to collectively determine the distances between individual nodes and the anchors, so that an (initial) position can be calculated. None of the alternatives engages in complicated calculations, so finding distance to anchors is communication bounded. Most of distributed localization algorithms share a common communication pattern: information is flooded into the network, starting at the anchor nodes. A network-wide flood by some anchor A is expensive since each node must forward A's information to its (potentially) unaware neighbors. This implies a scaling problem: flooding information from all anchors to all nodes will become too expensive for large networks, even with low anchor fractions. Fortunately a good position can still be derived with knowledge (position and distance) from a limited number of anchors. Therefore nodes can simply stop forwarding information when enough anchors have been $located$. This simple optimization (have a *flood limit*) has been proved to be highly effective in controlling the amount of communication.

There are several requirements an ad hoc positioning algorithm has to satisfy. Some of these requirement are similar to the requirements mentioned before. First, it has to be distributed: in a very large network of low memory and low bandwidth nodes, designed for intermittent operation, even shuttling the entire topology to a server in a hop by hop manner would put too high a strain on the nodes close to the base station or server. Partitioned areas would make centralization impossible, and anisotropic networks would put more strain on some nodes that have to support more forwarding traffic than others. Changing topologies would also make the centralized solution undesirable. Second, it has to minimize the amount of node to node communication and computation power, as the radio and the processor are the main sources of draining battery life. Also, it is desirable to have a low signaling complexity in the event a part of the network changes topology. Third, the positioning system should work even if the network becomes disconnected (in the context of sensor networks, the data can be later collected by a fly-over base station). Finally, our aim is to provide absolute positioning, in the global coordinate system of the GPS, as opposed to relative coordinates, for the following reasons: relative positioning might incur a higher signaling cost in the case the network topology changes, and absolute positioning enables a unique name-space, that of GPS coordinates.

Sum-dist: This method is also known as *DV-distance*[26]. The most simple solution for determining the distance to the anchors is simply adding the ranges encountered at each hop during the network flood. Sum-dist starts at the anchors, who send a message including their identity, position, and a path length set to 0. Each receiving node adds the measured range to the path length and forwards (broadcasts) the message if the flood limit allows it to do so. Another constraint is that when the node has received information about the particular anchor before, it is only allowed to forward the message if the current path length is less than the previous one. The end result is that each node will have stored the position and minimum path length to at least flood limit anchors.

DV-hop: A drawback of Sum-dist is that range errors accumulate when distance information is propagated over multiple hops. This cumulative error becomes significant for large networks with few anchors (long paths) and/or poor ranging hardware. A robust alternative is to use topological information by counting the number of hops instead of summing the (erroneous) ranges. This approach was named DV-hop by Niculescu and Nath [26], and Hop-TERRAIN by Savarese et al. [32]. DV-Hop propagation method is the most basic scheme, and it first employs a classical distance vector exchange so that all nodes in the network get distances, in hops, to the landmarks. DV-hop essentially consists of two flood waves. After the first wave, which is similar to Sum-dist, nodes have obtained the position and minimum hop count to at least flood limit anchors. The second calibration wave is needed to convert hop counts into distances such that nodes can compute a position. This conversion consists of multiplying the hop count with an average hop distance. Whenever an anchor A_1 infers the position of another anchor A_2 during the first wave, it computes the distance between them, and divides that by the number of hops to derive the average hop distance between A_1 and A_2. When calibrating, an anchor takes all remote anchors into account that it is aware of. Nodes forward (broadcast) calibration messages only from the first anchor that calibrates them, which reduces the total number of messages in the network.

When using DV-hop method, each node maintains a table of entries (X_i, Y_i, h_i) and exchanges updates only with its neighbors. Once a landmark gets distances to other landmarks, it estimates average size for one hop, which is then deployed as a correction to the entire network. When receiving the correction, a node may then have estimated distances to landmarks, in meters, which can be used to perform the triangulation. The correction a landmark (X_i, Y_i) computes is

$$c_i = \frac{\sum \sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2}}{\sum h_i}, i \neq j, \ for\ all\ landmarks\ j$$

A regular node gets an update from one of the landmarks, and it is usually the closest one, depending on the deployment policy and the time the correction phase of APS starts at each landmark. Corrections are distributed by controlled flooding, meaning that once a node gets and forwards a correction, it will drop all the subsequent ones. This policy ensures that most nodes will receive only one correction, from the closest landmark. When networks are large, a method to reduce signaling would be to set a TTL field for propagation packets, which would limit the number of landmarks acquired by a node. Here, controlled flooding helps keeping the corrections localized in the neighborhood of the landmarks they were generated from, thus accounting for non-isotropies across the network.

The advantages of the ”DV-hop” propagation scheme are its simplicity and the fact that it does not depend on measurement error. The drawbacks are that it will only work for isotropic networks, that is, when the properties of the graph are the same in all directions, so that the corrections that are deployed reason-arbitrary ably estimate the distances between hops.

Euclidean: A drawback of DV-hop is that it fails for highly irregular network topologies, where the variance in actual hop distances is very large. Niculescu and Nath [26] have proposed another method, named Euclidean, which is based on the local geometry of the nodes around an anchor. Again anchors initiate a flood, but forwarding the distance is more complicated than in the previous cases. When a node has received messages from two neighbors that know their distance to the anchor, and to each other, it can calculate the distance to the anchor. Figure 1.7 shows a node X that has two neighbors n_1 and n_2 with distance estimates (a and b) to an anchor. Together with the known ranges c, d, and e, there are two possible values (r_1 and r_2) for the distance of the node to the anchor. Niculescu describes two methods to decide on which, if any, distance to use. The neighbor vote method can be applied if there exists a third neighbor n_3 that has a distance estimate to the anchor and that is connected to either n_1 or n_2. Replacing n_2 (or n_1) by n_3 will again yield a pair of distance estimates. The correct distance is part of both pairs, and is selected by a simple voting. Of course, more neighbors can be included to make the selection more accurate. The second selection method is called common neighbor and can be applied if node n_3 is connected to both n_1 and n_2. Basic geometric reasoning leads to the conclusion that the anchor and n_3 are on the same or opposite side of the mirroring line n_1n_2, and similarly whether or not X and n_3 are on the same side. From this it follows whether or not X and the anchor lay on the same side. To handle the uncertainty introduced by range errors Niculescu implements a safety mechanism that rejects ill-formed (flat) triangles, which can easily derail the selection process by *neighbor vote* and *common neighbor*. This check verifies that the sum of the two smallest sides exceeds the largest side multiplied by a threshold, which is set to two times the range variance. For example, the triangle $\triangle Xn_1n_2$ in Figure 1.7 is accepted when $c + d > (1 + 2 \times RangeVar)$. Note that the safety check becomes more strict as the range variance (RangeVar) increases. This leads to a lower coverage, defined as the percentage of non-anchor nodes for which a position was determined.

We now describe some modifications proposed in [21] to Niculescu's *neighbor vote* method that remedy the poor selection of the location for X in important corner cases. The first problem occurs when the two votes are identical because, for instance, the three neighbors (n_1, n_2, and n_3) are collinear. In these cases it is hard to select the right alternative. Solution is to leave equal vote cases unsolved, instead of picking an alternative and propagating an error with 50% chance, then filter all indecisive cases by adding the requirement that the standard deviation of the votes for the selected distance must be at most $1/3$ of the standard deviation of the other distance. The second problem that is that of a bad neighbor with inaccurate information spoiling the selection process by voting for two wrong distances. This case is filtered out by requiring that the standard deviation of the selected distance is at most 5% of that distance. To achieve good coverage, they use both methods. If both produce a result, they use the result from the modified *neighbor vote* because they found it to be the most accurate of the two. If both fail, the flooding process stops leading to the situation where certain nodes are not able to establish the distance to enough anchor

nodes. Sum-dist and DV-hop, on the other hand, never fail to propagate the distance and hop count, respectively.

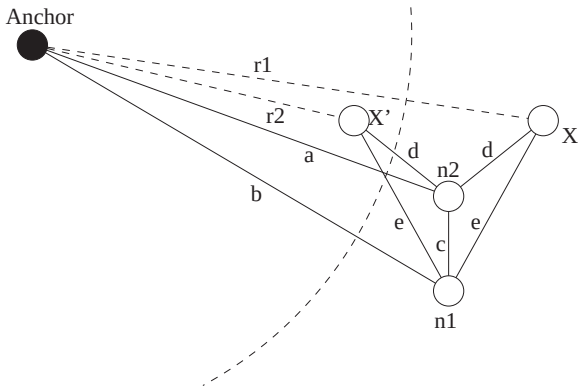


Fig. 1.7 Determining distance using Euclidean.

Let us see another example. An arbitrary node A needs to have at least two neighbors B and C which have estimates for the landmark L (See Figure 1.8). A also has measured estimates of distances for AB, AC, and BC, so there is the condition that: B and C, besides being neighbors of A, are neighbors of each other, or A knows distance BC, from being able to map all its neighbors in a local coordinate system. In any case, for the quadrilateral $ABCL$, all the sides are known, and one of the diagonals, which is BC in this case, is also known. This allows node A to compute the second diagonal AL, which in fact is the Euclidean distance from A to the landmark L. It is possible that A is on the same side of BC as L, shown as A' in the Figure 1.8 in which case the distance to L is different. The choice between the two possibilities is made locally by A either by voting, when A has several pairs of immediate neighbors with estimates for L, or by examining relation with other common neighbors of B and C. If it cannot be chosen clearly between A and A', an estimate distance to L won't be available for A until either more neighbors have estimates for L that will suit voting, or more second hop neighbors have estimates for L, so a clear choice can be made. Once the proper choice for A is available, the actual estimate is obtained by applying Pithagora's generalized theorem in triangles ACB, BCL, and ACL, to find the length of AL. An error reduction improvement applicable for the "Euclidean" propagation, but not for the "DV based" methods is for a landmark to correct all the estimates it forwards. It uses the true, GPS obtained coordinates, instead of relying on the measurement based received values.

1.3.1.4 Node Position Now nodes can determine their position based on the distance estimates to a number of anchors provided by one of the three alternatives (Sum-dist, DV-hop, or Euclidean). The determination of the node positions does not involve additional communication.

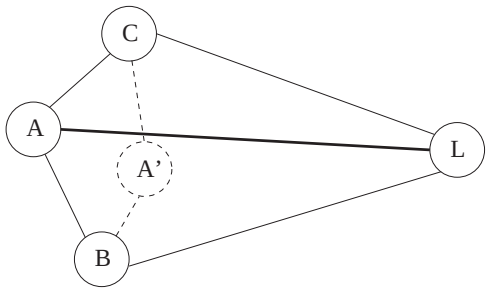


Fig. 1.8 Euclidean Propagation Method.

Lateration: The most common method for deriving a position is Lateration, which is a form of triangulation (introduced earlier in section 1.2.1). From the estimated distances d_i and known positions (x_i, y_i) of the anchors we derive the following system of equations:

$$\begin{aligned}(x_1 - x)^2 + (y_1 - y)^2 &= {d_1}^2,\\ (x_2 - x)^2 + (y_2 - y)^2 &= {d_1}^2,\\ &\vdots\\ (x_n - x)^2 + (y_n - y)^2 &= {d_n}^2,\end{aligned}$$

Where the unknown position is denoted by (x, y). The system can be linearized by subtracting the last equation from the first $n - 1$ equations.

$$\begin{aligned}{x_1}^2 - {x_n}^2 - 2(x_1 - x_n)x + {y_1}^2 - {y_n}^2 - 2(y_1 - y_n)y &= {d_1}^2 - {d_n}^2,\\ &\vdots\\ {x_{n-1}}^2 - {x_n}^2 - 2(x_{n-1} - x_n)x + {y_{n-1}}^2 - {y_n}^2 - 2(y_{n-1} - y_n)y &= {d_{n-1}}^2 - {d_n}^2,\end{aligned}$$

Reordering the terms gives a proper system of linear equations in the form $Ax = b$, where

$$A = \begin{bmatrix} 2(x_1 - x_n) & 2(y_1 - y_n) \\ \vdots & \vdots \\ 2(x_{n-1} - x_n) & 2(y_{n-1} - y_n) \end{bmatrix},$$

$$b = \begin{bmatrix} {x_1}^2 - {x_n}^2 + {y_1}^2 - {y_n}^2 + {d_n}^2 - {d_1}^2 \\ \vdots \\ {x_{n-1}}^2 - {x_n}^2 + {y_{n-1}}^2 - {y_n}^2 + {d_n}^2 - {d_{n-1}}^2 \end{bmatrix}.$$

The system is solved using a standard least-squares approach: $\hat{x} = (A^T A)^{-1} A^T b$. In exceptional cases the matrix inverse cannot be computed and Lateration fails. In

the majority of the cases, however, we succeed in computing a location estimate $\hat{x}$.We run an additional sanity check by computing the residue between the given distances d_i and the distances to the location estimate $\hat{x}$

$$residue = \frac{\Sigma_{i=1}^{n} \sqrt{(x_i - \hat{x})^2 + (y_i - \hat{y})^2} - d_i}{n}$$

A large residue signals an inconsistent set of equations; we reject the location $\hat{x}$ when the length of the residue exceeds the radio range.

Min-max: Lateration is quite expensive in the number of floating point operations that is required. A much simpler method is presented by Savvides et al. [33] as part of the N-hop multilateration approach.

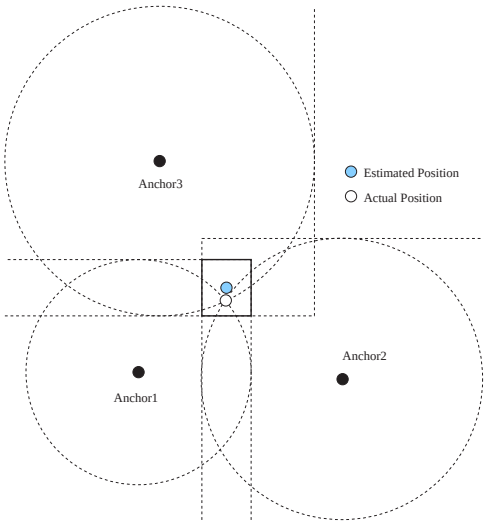


Fig. 1.9 Determining position using Min-Max.

The main idea is to construct a bounding box for each anchor using its position and distance estimate, and then to determine the intersection of these boxes. The position of the node is set to the center of the intersection box. Figure 1.9 illustrates the Minmax method for a node with distance estimates to three anchors. Note that the estimated position by Minmax is close to the true position computed through Lateration (i.e., the intersection of the three circles). The bounding box of anchor a is created by adding and subtracting the estimated distance d_a from the anchor position $(x_a, y_a) : [x_a - d_a, y_a - d_a] \times [x_a + d_a, y_a + d_a]$. The intersection of the bounding boxes is computed by taking the maximum of all coordinate minimums and the minimum of all maximums:$[max(x_i - d_i), max(y_i - d_i)] \times [min(x_i + d_i), min(y_i + d_i)]$. The final position is set to the average of both corner coordinates. As for Lateration, we only accept the final position if the residue is small.

1.3.1.5 Refinement Now we can refine the (initial) node positions computed so far. These positions are not very accurate, even under good conditions (high connectivity, small range errors), because not all available information is used in the first two phases. In particular, most ranges between neighboring nodes are neglected when the distances between node and anchor are determined. The iterative refinement procedure proposed by Savarese et al. [32] does take into account all internode ranges, when nodes update their positions in a small number of steps. At the beginning of each step a node broadcasts its position estimate, receives the positions and corresponding range estimates from its neighbors, and performs the Lateration procedure to determine its new position. In many cases the constraints imposed by the distances to the neighboring locations will force the new position towards the true position of the node. When, after a number of iterations, the position update becomes small, refinement stops and reports the final position.

The basic iterative refinement procedure outlined above may be too simple to be used in practice. The main problem is that errors propagate quickly through the network; a single error introduced by some node needs only d iterations to affect all nodes, where d is the network diameter. This effect was countered by (1) clipping undetermined nodes with non-overlapping paths to less than three anchors, (2) filtering out difficult symmetric topologies, and (3) associating a confidence metric with each node and using them in a weighted least-squares solution $(wAx = wb)$. The details (see [32]) are beyond the scope of this chapter, but the adjustments considerably improved the performance of the refinement procedure. This is largely due to the confidence metric, which allows filtering of bad nodes, thus increasing the (average) accuracy at the expense of coverage. The N-hop multilateration approach by Savvides et al. [33] also includes an iterative refinement procedure, but it is less sophisticated than the refinement discussed above. In particular, they do not use weights, but simply group nodes into so-called computation subtrees (over-constrained configurations) and enforce nodes within a subtree to execute their position refinement in turn in a fixed sequence to enhance convergence to a pre-specified tolerance.

1.3.2 Time-based Positioning Scheme (TPS)

TPS is a time-based positioning scheme for outdoor wireless sensor networks. Many applications of outdoor sensor networks require knowledge of physical sensor positions. For example, target detection and tracking is usually associated with location information. Further, knowledge of sensor location can be used to facilitate network functions such as packet routing and collaborative signal processing. Sensor position can also serve as a unique node identifier, making it unnecessary for each sensor to have a unique ID assigned prior to its deployment.

TPS relies on RF signal, which performs well compared to ultrasound, infrared, etc., in outdoor environments. They measure the difference in arrival times (TDoA) of beacon signals. In previous research, Time-of-Arrival (ToA) has proven more useful than RSSI in location determination. TPS does not need the specialized antennae generally required by an Angle-of-Arrival (AoA) positioning system. This time-

based location detection scheme avoids the drawbacks of many existing systems for outdoor sensor location detection. Compared to existing schemes proposed in the context of outdoor sensor networks, TPS scheme has the following characteristics and advantages:

- Time synchronization of all base stations and nodes is not required in TPS. Sensors measure the difference in signal arrival times using a local clock. Base stations schedule their transmissions based on receipt of other beacon transmissions and do not require synchronized clocks. Many existing location discovery systems for sensor networks require time synchronization among base stations, or between satellites and sensors. Imperfect time synchronization can degrade the positioning accuracy.

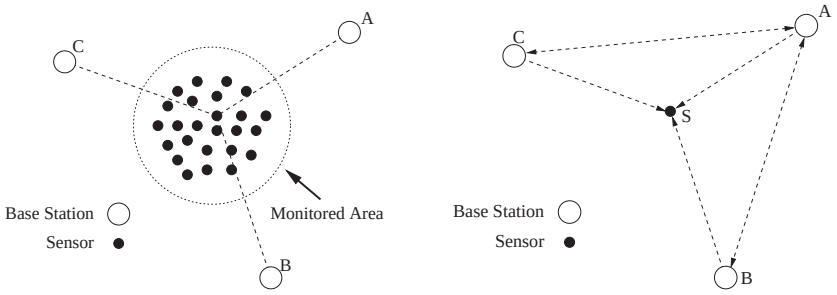
- There are no requirements for an ultrasound receiver, second radio or specialized antennae at base stations or sensors. TPS scheme does not incur the complexity, power consumption and cost associated with these components. (TPS sensors do require the ability to measure the difference in signal arrival times with precision.)

- TPS algorithm is not iterative and does not require a complicated refinement step. We refine position estimates by averaging time difference measurements over several beacon intervals prior to calculating position. This is useful to mitigate the effects of momentary interference and fast fading. This averaging requires less computation than repeatedly solving linear system matrices, least squares or multilateration algorithms.

- TPS has low computation cost. TPS location detection algorithm is based on simple algebraic operations on scalar values. On the other hand, multilateration based systems require matrix operations to optimize the objective functions (minimum mean square estimation or maximum likelihood estimation), which induces higher computation overhead at each sensor.

- Sensors listen passively and are not required to make radio transmissions. Base stations transmit all the beacon signals. This conserves sensor energy and reduces RF channel use. Connectivity based systems often require global flooding or global connectivity information to estimate range.

1.3.2.1 Network Model Assume that the sensors are deployed randomly over a 2-dimensional monitored area (on the ground). Each sensor has limited resources (battery, CPU, etc), and is equipped with an omni-directional antenna. Three base stations A, B, C, with known coordinates (x_a, y_a), (x_b, y_b), and (x_c, y_c), respectively, are placed beyond the boundary of the monitored area, as shown in Figure 1.10. Let us assume A be the master base station. Assume the monitored area is enclosed within the angle $\angle BAC$. Let the unknown coordinates of a sensor be (x, y),



(a) An example sensor network (b) Node S determines its position.

Fig. 1.10 TPS example

which will be determined by TPS. Each base station can reach all sensors in the monitored area. One restriction on the placement of these base stations is that they must be non-collinear, as otherwise, the sensor locations will be indistinguishable.

Note that these base stations will transmit RF beacon signals periodically to assist each sensor with location discovery. They have long-term power supplies and can receive RF signals from each other. Note that there is no time synchronization among these three base stations. However, TPS system requires base stations to detect signal arrival times with precision and to accurately calculate total turnaround delay. This calculated turn-around delay consists of a random delay combined with known system transmission and reception delays. If the monitored area is so large that 3 base stations can not cover the whole area completely, we can always divide the area into smaller subareas and place more base stations.

1.3.2.2 Positioning Scheme TPS time-based location detection scheme consists of two steps. The first step detects the time difference of signal arrival times from three base stations. These time differences are transformed into range differences from the sensor to the base stations. In the second step, we perform trilateration to transform these range estimates into coordinates.

Given the locations (x_a, y_a), (x_b, y_b), and (x_c, y_c) of base stations A, B, and C, respectively, TPS system determines the location (x, y) of sensor S, as shown in Figure 1.10.

- **Range Detection:** Let A be the master base station, which will initiate a beacon signal every T seconds. Each beacon interval begins when A transmits a beacon signal. Sensor S, base stations B and C will all receive As beacon signal respectively. B will reply to A with a beacon signal conveying the difference between the time the signal from A was received and the time the replay was sent. This signal will reach S. After receiving beacon signals from both A and B,C will reply to A with a beacon signal conveying the difference between the time the signal from A was received by C and the time the replay was sent. This signal will also reach S. Based on triangle inequality,

- **Location Computation:** Node S know the time the signal was sent from A and the time it was received by B, C and also by itself. Node S also has the same information about the signal sent by B and C. Now node S can calculate its position using trilateration.

1.3.2.3 Sources of Errors There are three major sources of errors for TPS: the *receiver system delay*, the *wireless multipath fading channel*, and the *non-line-of-sight (NLOS) transmission*. The receiver system delay is the time duration from which the signal hits the receiver antenna until the signal is decoded accurately by the receiver. This time delay is determined by the receiver electronics. Usually it is constant or varies in very small scale when the receiver and the channel is free from interference. This system delay can be predetermined and be used to calibrate the measurements. If base stations B and C can provide precise a priori information on receiver system time delay, their effect will be negligible. The wireless multipath fading channel will greatly influence the location accuracy of any location detection system. Major factors influencing multipath fading include multipath propagation, speed of the receiver, speed of the surrounding objects, and the transmission signal bandwidth. Multipath propagation refers to the fact that a signal transmitted from the sender can follow a multiple number of propagation paths to the receiving antenna. In TPS system, the performance is not affected by the speed of the receivers since all sensors and base stations are stationary. However, a moving tank in the surrounding area can cause interference. There are two important characteristics of multipath signals. First, the multiple non-direct path signals will always arrive at the receiver antennae latter than the direct path signal, as they must travel a longer distance. Second, in LOS transmission model, non-direct multipath signals will normally be weaker than the direct path signal, as some signal power will be lost from scattering. If NLOS exists, the non-direct multipath signal may be stronger, as the direct path is hindered in some way. Based on these characteristics, scientists can always design more sensitive receivers to lock and track the direct path signal. TPS mitigates the effect of multipath fading by measuring TDoA over multiple beacon intervals. TDoA measurements have been very effective in fading channels, as many detrimental effects caused by multipath fading and processing delay can be cancelled. Another factor related to wireless channels that causes location detection errors is NLOS transmission. To mitigate NLOS effects, base stations can be placed well above the surrounding objects such that there are line-of-sight transmission paths among all base stations and from base stations to sensors.

1.3.3 GPS-less Low Cost Outdoor Localization for Very Small Devices

Wireless networks of sensors greatly extend our ability to monitor and control the physical world. The availability of microsensors and low power wireless communications enables the deployment of densely distributed sensor/actuator networks for a wide range of biological and environmental monitoring applications, from marine to

soil and atmospheric contexts. Networked sensors can collaborate and aggregate the huge amount of sensed data to provide continuous and spatially dense observation of biological, environmental and artificial systems. Applications include environmental monitoring in the water and soil, tagging small animals unobtrusively, or tagging small and light objects in a factory or hospital setting. Instrumenting the physical world, particularly for such applications, requires that the devices we use as sensor nodes be small, light, unobtrusive and un-tethered. This imposes substantial restrictions on the amount of hardware that can be placed on these devices.

GPS solves the problem of localization in outdoor environments for PC class nodes. However, for large networks of very small, cheap and low power devices, practical considerations such as size, form factor, cost and power constraints of the nodes preclude the use of GPS on all nodes. The GPS-less system [10] addresses the problem of localization for such devices, with the following design goals.

- *RF-based*: They focus on small nodes which have some kind of short-range radio frequency (RF) transceiver. The primary goal is to leverage this radio for localization, thereby eliminating the cost, power and size requirements of a GPS receiver.

- *Receiver based*: In order to scale well to large distributed networks, the responsibility for localization must lie with the receiver node that needs to be localized and not with the reference points.

- *Ad hoc*: In order to ease deployment, a solution that does not require preplanning or extensive infrastructure is desired.

- *Responsiveness*: We need to be able to localize within a fairly low response time.

- *Low Energy*: Small, un-tethered nodes have modest processing capabilities, and limited energy resources. If a device uses all of its energy localizing itself, it will have none left to perform its task. Therefore, we desire to minimize computation and message costs to reduce power consumption.

- *Adaptive Fidelity*: In addition, we want the accuracy of our localization algorithms to be adaptive to the granularity of available reference points.

This scheme uses an idealized radio model and proposes a simple connectivity based localization method for such devices in unconstrained outdoor environments. It leverages the inherent radio-frequency (RF) communications capabilities of these devices. A fixed number of nodes in the network with overlapping regions of coverage serve as *reference* points and transmit periodic beacon signals. Nodes use a simple connectivity metric to infer proximity to a given subset of these reference points and then localize themselves to the centroid of the selected (proximate) reference points.

1.3.3.1 Localization Algorithm We considered two approaches to engineer an RF-based localization system, based on measurements of received signal strength and connectivity respectively.The first approach for RF-based localization is to use measured signal strength of received beacon signals to estimate distance, as in the RADAR system [8], with an outdoor radio signal propagation model. We discarded this approach for several reasons relating to our short-range (10m) radios. First, signal strength at short ranges is subject to unpredictable variation due to fading, multipath, and interferences. It does not therefore correlate directly with distance. Moreover, short range does not allow much gain in density of reference points when considering signal strength. We have found an idealized radio model useful for predicting bounds on the quality of connectivity based localization. We chose this model because it was simple and easy to reason about mathematically. This section presents this idealized model. To our surprise, this model compares quite well to outdoor radio propagation in uncluttered environments as we explore in the next section. We make two assumptions in our idealized model:

- Perfect spherical radio propagation.
- Identical transmission range (power) for all radios.

Multiple nodes in the network with overlapping regions of coverage serve as reference points (labeled R_1 to R_n). They are situated at known positions, (X_1, Y_1) to (X_n, Y_n), that form a regular mesh and transmit periodic beacon signals every T seconds containing their respective positions. We assume that neighboring reference points can be synchronized so that their beacon signal transmissions do not overlap in time. Furthermore, in any time interval, each of the reference points would have transmitted exactly one beacon signal.

Each mobile node listens for a fixed time period t and collects all the beacon signals that it receives from various reference points. We characterize the information per reference point R_i by a *connectivity metric* defined as:

$$CM_i = \frac{Nrecv(i,t)}{Nsent(i,t)} \times 100$$

Where $Nrecv(i,t)$ is the number of beacons sent by R_i that have been received in time t, and $Nsent(i,t)$ is the number of beacons that have been sent by R_i. In order to improve the reliability of our connectivity metric in the presence of various radio propagation vagaries, we would like to base our metric on a sample of at least S packets, where S is the sample size, a tunable parameter of our method (i.e., $Nsent(i,t) = S$). Since we know T to be the time period between two successive beacon signal transmissions, we can set t, the receivers sampling time as:

$$t = (s + 1 - \epsilon)T \quad (0 < \epsilon \ll 1)$$

From the beacon signals that it receives, the receiver node infers proximity to a collection of reference points for which the respective connectivity metrics exceed a

certain threshold. We denote the collection of reference points by $R_{i1}, R_{i2}, \ldots, R_{ik}$. The receiver localizes itself to the region which coincides to the intersection of the connectivity regions of this set of reference points, which is defined by the centroid of these reference points.

$$(X, Y) = \left(\frac{X_{i1} + X_{i2} + \ldots + X_{ik}}{k}, \frac{Y_{i1} + Y_{i2} + \ldots + Y_{ik}}{k} \right)$$

1.3.4 Computational Complexity of Sensor Network Localization

The localization problem for sensor networks is to reconstruct the positions of all of the sensors in a network, given the distances between all pairs of sensors that are within some radius r of each other. In the past few years, many algorithms for solving the localization problem were proposed, without knowing the computational complexity of the problem. Aspnes et al [5] showed that no *polynomial-time* algorithm can solve this problem in the worst case, even for sets of distance pairs for which a unique solution exists, unless ***RP = NP***.

Although the designs of the previous schemes have demonstrated clever engineering ingenuity, and their effectiveness is evaluated through extensive simulations, the focus of these schemes is on algorithmic design, without knowing the fundamental computational complexity of the localization process. In sensor network localization, since only nodes who are within a communication range can measure their relative distances, the graphs formed by connecting each pair of nodes who can measure each others distance are better modeled as unit disk graphs. Such constraints could have the potential of allowing computationally efficient localization algorithms to be designed.

The localization problem considered here is to reconstruct the positions of a set of sensors given the distances between any pair of sensors that are within some unit disk radius r of each other. Some of the sensors may be beacons, sensors with known positions, but our impossibility results are not affected much by whether beacons are available. To avoid precision issues involving irrational distances, it is assumed that the input to the problem is presented with the distances squared. If we make the further assumption that all sensors have integer coordinates, all distances will be integers as well.

For the main result, we consider a decision version of the localization problem, which we call UNIT DISK GRAPH RECONSTRUCTION. This problem essentially asks if a particular graph with given edge lengths can be physically realized as a unit disk graph with a given disk radius in two dimensions. The input is a graph G where each edge uv of G is labeled with an integer $l_u v^2$, the square of its length, together with an integer r^2 that is the square of the radius of a unit disk. The output is *yes* or *no* depending on whether there exists a set of points in R^2 such that the distance between u and v is $l_u v$ whenever uv is an edge in G and exceeds r whenever uv is not an edge in G.

The main result, is that UNIT DISK GRAPH RECONSTRUCTION is *NP-hard*, based on a reduction from CIRCUIT SATISFIABILITY. The constructed graph for a circuit with m wires has $O(m^2)$ vertices and $O(m^2)$ edges, and the number of solutions to the resulting localization problem is equal to the number of satisfying assignments for the circuit. In each solution to the localization problem, the points can be placed at integer coordinates, and the entire graph fits in an $O(m)$-by-$O(m)$ rectangle, where the constants hidden by the asymptotic notation are small. The construction also permits a constant fraction of the nodes to be placed at known locations. Formally:

Theorem 1 *There is a polynomial-time reduction from CIRCUIT SATISFIABILITY to UNIT DISK GRAPH RECONSTRUCTION, in which there is a one-to-one correspondence between satisfying assignments to the circuit and solutions to the resulting localization problem.*

A consequence of this result is:

Corollary 2 *There is no efficient algorithm that solves the localization problem for sparse sensor networks in the worst case unless P = NP.*

It might appear that this result depends on the possibility of ambiguous reconstructions, where the position of some points is not fully determined by the known distances. However, if we allow randomized reconstruction algorithms, a similar result holds even for graphs that have unique reconstructions.

Corollary 3 *There is no efficient randomized algorithm that solves the localization problem for sparse sensor networks that have unique reconstructions unless RP = NP.*

Finally, because the graph constructed in the proof of Theorem 1 uses only points with integer coordinates, even an approximate solution that positions each point to within a distance $\epsilon < 1/2$ of its correct location can be used to find the exact locations of all points by rounding each coordinate to the nearest integer. Since the construction uses a fixed value for the unit disk radius r (the natural scale factor for the problem), we have:

Corollary 4 *The results of Corollary 2 and Corollary 3 continue to hold even for algorithms that return an approximate location for each point, provided the approximate location is within ϵr of the correct location, where ϵ is a fixed constant.*

What we do not know at present is whether these results continue to hold for solutions that have large positional errors but that give edge lengths close to those in the input. Our suspicion is that edge-length errors accumulate at most polynomially across the graph, but we have not yet carried out the error analysis necessary to prove this. If our suspicion is correct, we would have:

Conjecture 1 *The results of Corollary 2 and Corollary 3 continue to hold even for algorithms that return an approximate location for each point, provided the relative error in edge length for each edge is bounded by ϵ/n^c for some fixed constant c.*

1.4 TARGET TRACKING AND CLASSIFICATION

One of the most important areas where the advantages of sensor networks can be exploited is for tracking mobile targets. Scenarios where such network may be deployed can be both military (tracking enemy vehicles, detecting illegal border crossings) and civilian (tracking the movement of wild animals in wildlife preserves). Typically, for accuracy, two or more sensors are simultaneously required for tracking a single target, leading to coordination issues. Additionally, given the requirements to minimize the power consumption due to communication or other factors, we would like to select the bare essential number of sensors dedicated for the task while all other sensors should preferably be in the hibernation or off state. In order to simultaneously satisfy the requirements like power saving and improving overall efficiency, we need large scale coordination and other management operations. These tasks become even more challenging when one considers the random mobility of the targets and the resulting need to coordinate the assignment of the sensors best suited for tracking the target as a function of time. In this section managing and coordinating a sensor network for tracking moving targets is discussed.

The power limitation due to the small size of the sensors, the large numbers of sensors which need to be deployed and coordinated, and the ability to deploy sensors in an ad hoc manner give rise to a number of challenges in sensor networks. Each of these needs to be addressed by any proposed architecture in order for it to be realistic and practical.

- **Scalable Coordination:** A typical deployment scenario for a sensor network comprises of a large number of nodes reaching in the thousands to tens of thousands. At such large scales, it is not possible to attend to each node individually due to a number of factors. Sensors nodes may not be physically accessible, nodes may fail and new nodes may join the network. In such dynamic and unpredictable scenarios, scalable coordination and management functions are necessary which can ensure a robust operation of the network. In the light of target tracking, the coordination function should scale with the size of the network, the number of targets to be tracked, number of active queries etc.

- **Tracking Accuracy:** To be effective, the tracking system should be accurate and the likelihood of missing a target should be low. Additionally, the dynamic range of the system should be high while keeping the response latency, sensitivity to external noise and false alarms low. The overall architecture should also be robust against node failures.

- **Ad Hoc Deployability:** A powerful paradigm associated with sensor networks is their ability to be deployed in an ad hoc manner. Sensors may be thrown in

an area affected by a natural or man made disaster or air dropped to cover a geographical region. Thus sensor nodes should be capable of organizing themselves into a network and achieving the desired objective in the absence of any human intervention or fixed patterns in the deployment.

- **Computation and Communication Costs:** Any protocol being developed for sensor networks should keep in mind the costs associated with computations and communication. With current technology, the cost of computation locally is lower than that of communication in a power constrained scenario. As a consequence, emphasis should be put on minimizing the communication requirements.

- **Power Constraints:** The available power in each sensor is limited by the battery lifetime due to the difficulty or impossibility of recharging the nodes. As a consequence, protocols which tend to minimize the energy consumption or power aware protocols which adapt to the existing power levels are highly desirable. Additionally, efforts should be made to turn off the nodes themselves if possible in the absence of sensing or coordination operations.

1.4.1 Collaborative Signal Processing

Power consumption is a critical consideration in a wireless sensor network. The limited amount of energy stored at each node must support multiple functions, including sensor operations, on-board signal processing, and communication with neighboring nodes. Thus, one must consider power-efficient sensing modalities, low sampling rates, low-power signal processing algorithms, and efficient communication protocols to exchange information among nodes. To facilitate monitoring of a sensor field, including detection, classification,identification, and tracking of targets, global information in both space and time must be collected and analyzed over a specified space-time region. However, individual nodes only provide spatially local information. Furthermore, due to power limitation, temporal processing is feasible only over limited time periods. This necessitates *Collaborative Signal Processing (CSP)*(i.e., Collaboration between nodes to process the space-time signal). A CSP algorithm can benefit from the following desirable features:

- **Distributive processing:** Raw signals are sampled and processed at individual nodes but are not directly communicated over the wireless channel. Instead, each node extracts relevant summary statistics from the raw signal, which are typically of smaller size. The summary statistics are stored locally in individual nodes and may be transmitted to other nodes upon request.

- **Goal-oriented, on-demand processing:** To conserve energy, each node only performs signal processing tasks that are relevant to the current query. In the absence of a query, each node retreats into a standby mode to minimize energy consumption. Similarly, a sensor node does not automatically publish extracted information (i.e., it forwards such information only when needed.).

- **Information fusion:** To infer global information over a certain space-time region from local observations, CSP must facilitate efficient, hierarchical information fusion and progressively lower bandwidth information must be shared between nodes over progressively large regions. For example, (high bandwidth) time series data may be exchanged between neighboring nodes for classification purposes. However, lower bandwidth *CPA (closest point of approach)* data may be exchanged between more distant nodes for tracking purposes.

- **Multi-resolution processing:** Depending on the nature of the query, some CSP tasks may require higher spatial resolution involving a finer sampling of sensor nodes, or higher temporal resolution involving higher sampling rates. For example, reliable detection may be achievable with a relatively coarse space-time resolution, whereas classification typically requires processing at a higher resolution.

1.4.2 Target Tracking Using Space-Time Cells

1.4.2.1 Introduction Each object in a geographical region generates a time-varying space-time signature field that may be sensed in different modalities, such as acoustic, seismic or thermal. The nodes sample the signature field spatially and the density of nodes should be commensurate with the rate of spatial variation in the field. Similarly, the time series from each sensor should be sampled at a rate commensurate with the required bandwidth. Thus, the rate of change of the space-time signature field and the nature of the query determines the required space-time sampling rate. A moving object in a region corresponds to a peak in the spatial signal field that moves with time. Tracking an object corresponds to tracking the location of the spatial peak over time.

1.4.2.2 Using Space-Time Cells To enable tracking in a sensor network, the entire space-time region must be divided into *space-time cells* to facilitate local processing . The size of a space-time cell depends on the velocity of the moving target and the decay exponent of the sensing modality. It should approximately correspond to a region over which the space-time signature field remains nearly constant. In principle, the size of space-time cells may be dynamically adjusted as new space-time regions are created based on predicted locations of targets. Space-time signal averaging may be done over nodes in each cell to improve the signal to noise ratio. We note that the assumption of constant signature field over a space-time cell is at best an approximation in practice due to several factors, including variations in terrain, foliage, temperature gradients and non-isotropic nature of source signal. However, such an approximation may be judiciously applied in some scenarios for the purpose of reducing intra-sensor communication as well to improve algorithm performance against noise.

1.4.2.3 Single Target Tracking One of the key premises behind the networking algorithms being developed at Wisconsin [23] is that routing of information in a sensor network should be geographic-centric rather than node-centric. In other words, from the viewpoint of information routing, the geographic locations of the nodes are the critical quantities rather than their arbitrary identities. In the spirit of space-time cells, the geographic region of interest is divided into smaller regions (spatial cells) that facilitate communication over the sensor network. Some of the nodes in each cell are designated as *manager nodes* for coordinating signal processing and communication in that cell.

Figure 1.11 illustrates the basic idea of region-based CSP for detection and tracking of a single target. Under the assumption that a potential target may enter the monitored area via one of the four corners, four cells, A, B, C and D, are created by the UW-API protocols [23]. Nodes in each of the four cells are activated to detect potential targets.

Each activated node runs an energy detection algorithm whose output is sampled at an a priori fixed rate depending on the characteristics of expected targets. Suppose a target enters Cell A. Tracking of the target consists of the following five steps:

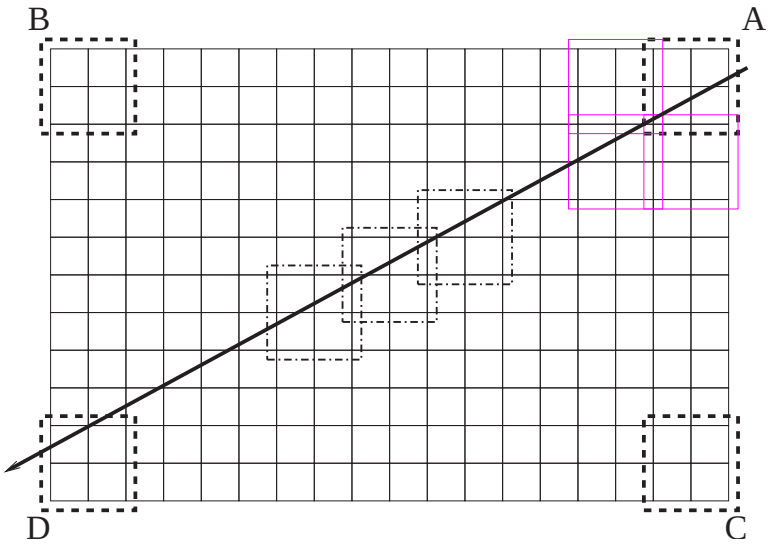


Fig. 1.11 A Schematic illustrating detection and tracking of a single target.

1. Some and perhaps all of the nodes in Cell A detect the target. These nodes are the active nodes and Cell A is the active cell. The active nodes also yield CPA time information. The active nodes report their energy detector outputs to the manager nodes at N successive time instants.

2. At each time instant, the manager nodes determine the location of the target from the energy detector outputs of the active nodes. The simplest estimate of

target location at an instant is the location of the node with the strongest signal at that instant. However, more sophisticated algorithms for target localization may be used. Such localization algorithms justify their higher complexity only if the accuracy of their location determination is finer than the node spacing.

3. The manager nodes use locations of the target at the N successive time instants to predict the location of the target at $M(< N)$ future time instants.

4. The predicted positions of the target are used by the UW-API protocols [23] to create new cells that the target is likely to enter. This is illustrated in Figure 1.11 where the three dotted cells represent the regions that the target is likely to enter after the current active cell (Cell A in Figure 1.11). A subset of these cells is activated by the UW-API protocols for subsequent detection and tracking of the target.

5. Once the target is detected in one of the new cells, it is designated as the new active cell and the nodes in the original active cell (Cell A in Figure 1.11) may be put in the standby state to conserve energy.

Steps 1 5 are repeated for the new active cell and this forms the basis of detecting and tracking a single target. For each detected target, an information field containing tracking information, such as the location of the target at certain past times, is usually passed from one active cell to the next one. This is particularly important in the case of multiple targets.

1.4.2.4 Multiple Target Tracking Figure 1.11 illustrates detection and tracking of a single target. If multiple targets are sufficiently separated in space or time, that is they occupy distinct space-time cells, essentially the same procedure as described in Section 1.4.2.3 may be used: a different track is initiated and maintained for each target. Sufficient separation in time means that the energy detector output of a particular sensor exhibits distinguishable peaks corresponding to the CPAs of the two targets. Similarly, sufficient separation in space means that at a given instant the spatial target signatures exhibit distinguishable peaks corresponding to nodes that are closest to the targets at that instant. The assumption of sufficient separation in space and/or time may be too restrictive in general. In such cases, classification algorithms are needed that operate on spatio-temporal target signatures to classify them. This necessarily requires a priori statistical knowledge of typical signatures for different target classes.

1.4.2.5 Target Classification Here, we focus on single-node (no collaboration between nodes) classification based on temporal target signatures: a time series segment is generated for each detected event at a node and processed for classification. Some form of temporal processing, such as a fast Fourier transform (FFT), is performed and the transformed vector is fed to a bank of classifiers corresponding to different target classes. The outputs of the classifiers that detect the target, active classifiers, are reported to the manager nodes as opposed to the energy detector

outputs. Steps (1) to (5) in Section 1.4.2.3 are repeated for all the active classifier outputs to generate and maintain tracks for different classified targets. In some cases, both energy-based CPA information and classifier outputs may be needed.

Now we briefly describe the three classifiers explored in this paper. Given a set of N-dimensional feature vectors $\{x; x \in R^N\}$, we assume that each of them is assigned a class label, $\omega_c \in \Omega = \{\omega_1, \omega_2, ..., \omega_m\}$, that belongs to a set of m elements. We denote by $p(\omega_c)$ the prior probability that a feature vector belongs to class ω_c. Similarly, $p(\omega_c|x)$ is the posterior probability for class ω_c given that x is observed.

A *minimum error classifier* maps each vector x to an element in. such that the probability of misclassification (i.e., the probability that the classifier label is different from the true label) is minimized. To achieve this minimum error rate, the optimal classifier decides x has label ω_i if $p(\omega_i|x)$ for all $j \neq i, \omega_i, \omega_j \in \Omega$. In practice, it is very difficult to evaluate the posterior probability in closed form. Instead, one may use an appropriate discriminant function $g_i(x)$ that satisfies $g_i(x) > g_j(x)$ if $p(\omega_i|x) > p(\omega_j|x)$ for $j \neq i$, for all x. Then minimum error classification can be achieved as: decide x has label ω_i if $g_i(x) > g_j(x)$ for $j \neq i$. The minimum probability of misclassification is also known as the *Bayes* error, and a minimum error classifier is also known as a Bayes classifier or a maximum a posterior probability (MAP) classifier. Below, we briefly discuss three classifiers that approximate the optimal Bayes classifier.

- ***k*-Nearest Neighbor (kNN) Classifier:** The kNN classifier uses all the training features as the set of prototypes $\{p_k\}$. During testing phase, the distance between each test vector and every prototype is calculated, and the k prototype vectors that are closest to the test vector are identified. The class labels of these k-nearest prototype vectors are then combined using majority vote or some other method to decide the class label of the test vector. When $k = 1$, the kNN classifier is called the nearest neighbor classifier. It is well-known ***[9]*** that asymptotically (in the number of training vectors), the probability of misclassification of a nearest neighbor classifier approaches twice the (optimal) Bayes error. Hence the performance of a nearest neighbor classifier can be used as a baseline to gauge the performance of other classifiers. However, as the number of prototypes increases, a kNN classifier is not very suitable for actual implementation since it requires too much memory storage and processing power for testing.

- **Maximum Likelihood Classifier:** Using Gaussian Mixture Density Model In this classifier, the distribution of training vectors from the same class is modeled as a mixture of Gaussian density functions. That is, the likelihood function is modeled as:

$$p(x|\omega_i) \propto G_i(\, x|\theta_i) = \sum_k |\Lambda_{ik}|^{-N/2} exp\left(-\frac{1}{2}(x - m_{ik})^T {\Lambda_{ik}}^{-1} (x - m_{ik})\right) \tag{1.1}$$

where $\theta_i = [m_{i1}, m_{i2}, ..., m_{ip}, \Lambda_{i1}, \Lambda_{i2}, ..., \Lambda_{ip}]$ are the mean and covariance matrix parameters of the P mixture densities corresponding to class ω_i. These model parameters can be identified by applying an appropriate clustering algorithm, such as the k-means algorithm, or the Expectation-Maximization algorithm to the training vectors of each class. The discriminant function is computed as $g_i(x) = G_i(x|\theta_i)p(\omega_i)$ where the prior probability $p(\omega_i)$ is approximated by the relative number of training vectors in class i. In the numerical examples The can also be modeled data as Gaussian rather than a Gaussian mixture $(P = 1)$. Furthermore, you can use the maximum likelihood (ML) classifier (uniform prior probabilities).

- **Support Vector Machine (SVM) Classifier:** A support vector machine (SVM) is essentially a linear classifier operating in a higher dimensional space. Consider a binary classification problem without loss of generality. Let $\{\varphi(x)\}_{i=1}^{M}$ be a set of nonlinear transformations mapping the N-dimensional input vector to an M-dimensional feature space $(M > N)$. A linear classifier, characterized by the weights $w_1, w_2, ..., w_M$, operates in this higher dimensional feature space $g(x) = \sum_{j=1}^{M} w_j \varphi_j(x) + b$, where b is the bias parameter of the classifier. The optimal weight vectors for this classifier can be represented in terms of a subset of training vectors, termed the support vectors $w_j = \sum_{i=1}^{Q} \alpha_i \varphi_j(x_i), j = 1, 2, ..., M$. Using the above representation for the weight vectors, the linear classifier can be expressed as $g(x) = \sum_{i=1}^{Q} \alpha_i K(x, x_i) + b$ where $K(x, x_i) = \sum_{j=1}^{M} \varphi_j(x)\varphi_j(x_i)$ is the symmetric kernel representing the SVM. In the numerical examples presented in this paper, we use a third degree polynomial kernel: $K(x, x_i) = (x^T x_i + 1)^3$. In practice, the SVM discriminant function $g(x)$ is computed using the kernel representation, bypassing the nonlinear transformation into the higher dimensional space [10]. The classifier design then corresponds to the choice of the kernel and the support vectors. By appropriately choosing the kernel, an SVM can realize a neural network classifier as well. Similar to neural networks, the training phase can take a long time. However, once the classifier is trained, its application is relatively easy. In general, a different SVM is trained for each class. The output of each SVM can then be regarded as an estimate of the posterior probability for that class and the MAP decision rule can be directly applied.

1.4.3 Target Tracking Based on Cooperative Binary Detection

1.4.3.1 Introduction Unlike other sensor network-based methods, which depend on determining distance to the target or the angle of arrival of the signal, cooperative tracking approach requires only that a sensor be able to determine if an object is somewhere within the maximum detection range of the sensor. Cooperative tracking is proposed as a method for tracking moving objects and extrapolating their paths in the short term. By combining data from neighboring sensors, this approach enables tracking with a resolution higher than that of the individual sensors being

used. In cooperative tracking, statistical estimation and approximation techniques can be employed to further increase the tracking precision, and enables the system to exploit the tradeoff between accuracy and timeliness of the results. This work focuses on acoustic tracking, however the presented methodology is applicable to any sensing modality where the sensing range is relatively uniform.

Cooperative tracking is a solution for tracking objects using sensor networks, and may achieve a high degree of precision while meeting the constraints of sensor network systems. The approach uses distributed sensing to identify an object and determine its approximate position, and local coordination and processing of sensor data to further refine the position estimate. The salient characteristics of the cooperative tracking approach are that it achieves resolution that is finer than that of the individual sensors being used and that it provides early estimates of the objects position and velocity. Thus cooperative tracking is useful for short-term extrapolation of the objects path. Here an acoustic tracking system for wireless sensor networks is considered as a practical application of the cooperative tracking methodology. Acoustic tracking relies on a network of microphone-equipped sensor nodes to track an object by its characteristic *acoustic signature*.

1.4.3.2 Model In the real world, objects can move arbitrarily, i.e. possibly changing speed and direction at any time. The representation of such arbitrary paths may be cumbersome, and unnecessarily complex for the purpose of tracking the objects path with a reasonable degree of precision. Instead, an approximation of the path can be considered. Cooperative tracking uses *piecewise linear approximation* to represent the path of the tracked object. Although the object itself may move arbitrarily, its path is considered as a sequence of line segments along which the object moves with a constant speed. The degree to which the actual path diverges from its representation depends on several factors, including speed and turning radius of the object itself. For vehicles such as cars driving along highways the difference is quite small, whereas for a person walking a curved route with tight turns it may be significant. In either case, accuracy can be improved by increasing the resolution of the sensor network, either through increasing sensor density or by other means.

In cooperative tracking, it is assumed each node is equipped with a sensor (in the case of acoustic tracking, a microphone) and a radio for communication with nearby nodes. Since these embedded systems are designed to be small and cheap, the sensors they are equipped with are unlikely to be very sophisticated. Traditionally, tracking relies on sensors that are long range and can detect the direction of an object and the distance to it. This is not the case with sensor networks: the microphones used for acoustic tracking are likely to be short range, non-directional and poorly suited for detecting the distance to the sound source. The method presented here assumes that only binary (on-off) detection can be used. It is possible to generalize this analysis if multilevel detection is feasible. Moreover, without proper calibration the detection range may be neither uniform nor exact. Figure 1.12 shows the model of a sensor considered in this paper. Given a sensor with a nominal (non-calibrated) range R, the object will always be detected if it is distance $R - e$ or less away from the sensor,

detected some of the time between $R - e$ and $R + e$, and never detected beyond that range. We found that setting $e = 0.1R$ comes fairly close to the actual behavior of the sensors used in our experiments.

To track an object, it must be identified and its presence detected. For acoustic tracking, objects are identified based on their acoustic signature, which is a characteristic sound pattern or a set of frequencies unique to that object. For simplicity, It is assumed that the object emits sound of a frequency not present in the environment, so there are no false positives. However, the results are fairly robust with respect to intermittent detection (false negatives) during the period of observations.

It is worth noting that the sensor model is generic enough to encompass other sensing modalities beyond acoustic. All that is required is a sensor with a relatively uniform range, as defined above, which is capable of differentiating the target from the environment. Magnetometer, a device that detects changes in magnetic fields, is one such sensor.

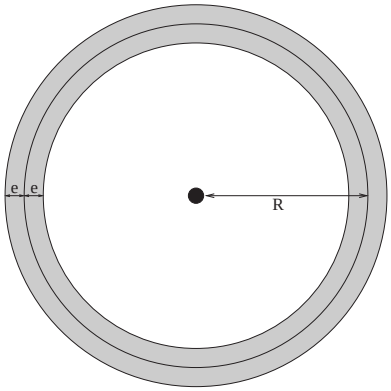


Fig. 1.12 Model of a sensor. For nominal sensing range R, the object is always detected when it is $R - e$ away or closer, never detected beyond $R + e$, and has a non-negative chance of detection between $R - e$ and $R + e$.

1.4.3.3 Algorithm The simplest distributed tracking algorithm entails simply recording the times when each sensor detects the object and then performing line fitting on the resulting set of points. While simple, this approach is not very precise: it can only track the object with a resolution of the sensor range R. Moreover, if a sensor detects the object more than once as it moves through the sensors detection range, that information is lost.

The position of a stationary object, or a moving object for that matter, which is determined using this method is not very precise and depends heavily on the number, the detection range and precision of sensors that detect the sound. Instead of looking at a single position measurement, we are interested in the path of a moving object, which is a sequence of positions over a period of time. Combining a large number of somewhat imprecise position estimates distributed over space and time may

yield surprisingly accurate results. Cooperative tracking addresses the problem of high-resolution tracking using sensor networks. It improves accuracy by combining information from neighboring sensors. The only requirement for cooperative tracking to be used is that the density of sensor nodes must be high enough for the sensing ranges of several sensors to overlap. When the object of interest enters the region where multiple sensors can detect it, its position can be pinned down with a higher degree of accuracy, since the intersection area is smaller than the detection area of a single node. The outline of a generic cooperative tracking algorithm is as follows:

- Each node records the duration for which the object is in its range.
- Neighboring nodes exchange these times and their locations.
- For each point in time, the objects estimated position is computed as a weighted average of the detecting nodes locations.
- A line fitting algorithm is run on the resulting set of points.

Several of these steps require careful consideration. First, the algorithm implicitly assumes that the nodes clocks are synchronized, and that the nodes know their locations. Second, we obtain a position reading by a weighted average of the locations of the nodes that detected the sound at a given instant, but the exact weighting scheme is not specified. This is an important issue, as selecting an appropriate scheme will improve accuracy, while a poor choice might be detrimental to it.

The simplest choice is to assign equal weights to all sensors readings. This effectively puts the estimate of the objects position at the centroid of the polygon with sensors acting as vertices. This is a safe choice, and intuitively it should be more accurate than non-cooperative tracking. However it is possible to do even better. Consider Figure 1.13: sensors that are closer to the path of the target will stay in sensor range for a longer duration. Thus to increase accuracy, the weight of a sensors reading should be proportional to some function of the duration for which the target has been in range of that sensor.

Once the individual position estimates are computed, the final step of the line fitting algorithm can begin. Least squares regression can be used to find the equation of the line. It is interesting to note that the duration-based weighting scheme for position estimates moves the points closer to the actual path, thus reducing variance in the least squares computation. Also important is the fact that the multi-step approach enables early estimates of the path to be computed, so that continuous refinement is possible as more data points become available. The resulting equation of the line extrapolates the path of the object until it changes course sharply. This information may be used by the system, e.g. for asynchronous wake up of nodes likely to be in its path.

1.4.3.4 Data Aggregation The final step of the algorithm involves performing a line fitting computation on the set of all the position estimates (or some subset of them). Unlike position estimates, which can be performed in a distributed

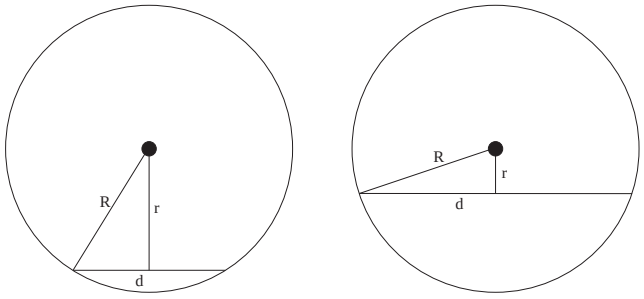


Fig. 1.13 If the object's speed is constant, detection time is directly proportional to path segment d and inversely proportional to distance r from the sensor to object's trajectory.

manner with only local communication, this necessitates collecting sensor readings from many sensor nodes at a centralized location for processing. This process is called *data aggregation*, and it is present in one form or another in virtually all sensor network applications. The main concerns for data aggregation are timeliness and resource usage. Timeliness, with respect to sensor data, is critical to real-time monitoring and control applications where stale data is useless or even detrimental. Resources, in particular network bandwidth and message buffers, are quite scarce in networked embedded systems. Low bandwidth of small wireless transmitters and the potential for contention with other messages drastically limit the amount of data that can pass through the network.

We assume that some nodes in the sensor network are gateways nodes connected to outside networks such as the Internet. To process the data from the sensor network, it needs to be sent through one of these gateway nodes to the more powerful computers connected to the outside network. To do this efficiently, a tree rooted at each gateway is constructed and spanning the entire network. Each sensor node in the tree collects data from its children and sends it up the tree to either the closest or the least busy gateway. This scheme addresses the conflicting requirements of low bandwidth usage and timeliness of data: a near-shortest path is always taken, unless its links are overloaded. What mentioned above is practical only if the outside network is low latency and high bandwidth, so it does not matter to which gateway the data is sent.

1.4.4 Distributed Prediction Tracking (DPT)

The Distributed Prediction Tracking (DPT) algorithm is specifically aimed at addressing the various challenges outlined in the Section 1.4 while accurately tracking moving targets. As the name suggests, this algorithm does not require any central control point, eliminating the possibility of a single point of failure and making it robust against random node failures. The tracking task is carried out distributively by sequentially involving the sensors located along the track of the moving target. DPT

assumes a cluster based architecture for the sensor network and the choice was motivated by the need to ensure the sensor networks scalability and energy efficiency. Any suitable clustering mechanism from those proposed in literature may be used and note that DPT does not impose any specific requirements or restrictions on the choice of the clustering algorithm.

1.4.4.1 Assumptions of the DPT Algorithm While no assumption is made on the choice of the clustering algorithm, we assume that the Cluster Head *(CH)* has the following information about all sensors belonging to its cluster: (1) Sensor identity, (2) Location and (3) Energy level. When tracking a moving target and deciding which sensors to use for tracking, the cluster heads decision-making procedure will be based on this information. The assumptions about the sensors are enumerated below. These assumptions are realistic and targeted at reducing the energy cost and prolonging the whole networks lifetime as well.

1. All sensors have the same characteristics.

2. Sensors are randomly distributed across the whole sensing area with uniform density.

3. Each sensor has two sensing radii, normal beam r and high beam R. The default operation uses the low beam and the high beam is turned on only when necessary. The following relationship holds between the energy consumed by the low and high beams:

$$\frac{E_{lowbeam}}{E_{highbeam}} = \frac{r^2}{R^2} \tag{1.2}$$

4. A sensors communication and sensing channel stay in the hibernation mode most of the time where they consume minimal energy. The communication channel will wake up routinely to receive possible messages from its cluster head. The sensor will perform sensing according to its cluster heads requirements.

In order to produce accurate enough information to locate the moving target, DPT requires that at any given time there should be at least 3 sensors to sense the target jointly. The number 3 is chosen as the compromise between increasing accuracy and minimizing the consumed energy (note that this is not a hard assumption and depending on the sensor node specifications the number may vary).

No specific assumptions are made about the movement pattern of the targets. However, DPT assumes that the targets originate outside the sensing region and then move inside. Also, it is assumed that the movement of each tracked target needs to be forwarded to a central location which we term the sink. In reality, the sink could be either a special node or a terminal associated with a human.

1.4.4.2 DPT Algorithm The DPT algorithm comes into play after sensors are deployed and clusters are formed. DPT distinguishes between the *border sensors*, sensors located within a given distance of the border, and *non border sensors* in terms of their operation. While border sensors are required to keep sensing all times in order to detect all targets that enter the sensing region, the non-border sensors sensing channel hibernates unless it is specifically asked to sense by its cluster head. Since the target is assumed to move from outside into the sensing area, it will be detected by the border sensors when trespassing the border. As soon as a target is detected, a sequence of tasks in the order of *sense-predict-communicate-sense* are carried out distributively by a series of sensors that are located along the targets track. This forms the essential idea behind the DPT algorithm.

Let CH_1,CH_2,...,CH_N denote the sequence of cluster heads that become involved with tracking the targets as it proceeds from its very first location to the last. The information gathered by each cluster head is sent all the way back to the sink (either sent intact or after being aggregated) for further processing as well as to the downstream cluster head CH_{i+1}. The *Target Identity* is created when the target is first detected. This identity is unique and all cluster heads that co-track this target use it to identify the Target. In order to facilitate the smooth tracking of the target, CH_i predicts the future location of the moving target, and informs the downstream cluster head CH_{i+1} ahead of time about this target. The accuracy of the *prediction* is very important if downstream cluster heads are to be identified accurately and the overall tracking mechanism is to be effective. Many prediction mechanisms are possible, the simplest one is a linear predictor, which only uses the previous two locations to linearly predict the third location. Higher order prediction can also be adopt, which predicts the n^{th} location information based on previous $n-1$ actual locations. Higher order prediction results in more accurate results, though, at the cost of greater energy consumption.

Sensor Selection Algorithm: After cluster head CH_i predicts the location of the target, the downstream cluster head CH_{i+1} towards which the target is headed receives a message from CH_i indicating this predicted location. With information of all sensors belonging to CH_{i+1} available in its database, the search algorithm running at CH_{i+1} is able to locally decide the sensor-triplet to sense the target. The selection rule chooses 3 sensors (if possible) such that their distances to the predicted location are not only less than the sensors normal beam r, but also the smallest. After the sensor triplet is chosen, CH_{i+1} sends them a *wake-up* message so that they are ready to sense the target. If the prediction and selections process succeeds, after sensing, each sensor will send a location message to CH_{i+1}. If CH_{i+1} in unable to find enough sensors eligible for this sensing task with the normal sensing beam, it will try to search for eligible sensors within a distance R, the higher sensing beam, from the predicted location. The selected sensors, whose distance from the predicted location is greater than r and lower than R, will now be contacted and instructed to sense with their high beam, while the rest of the sensors in the triplet use their normal beam. If CH_{i+1} is unable to find enough sensors even with high sensing beams, it asks its neighboring cluster heads for help.

Failure Recovery: Let's first identify two possible failure scenarios. As described in the previous subsections, each upstream cluster head sends a message to the expected downstream cluster head. If the upstream cluster head does not get any confirmation from the downstream cluster head after a given period of time, then it assumes that the downstream cluster head is no longer available and the target has been lost. Another failure scenario occurs when the target changes it direction or speed so abruptly that it moves significantly away from the predicted location and falls out of the detectable region of the sensor-triplet selected for the sensing task. In both of these failure scenarios a straight forward solution is to wake up all sensors within a given area, which is calculated based on the target's previous actual location. The *re-capture* radius σ is an important parameter in this process and is decided by the target's moving speed and time elapsed since it was last sensed.

1.5 EXPERIMENTAL LOCATION AND TRACKING SYSTEMS

In this section, several location systems are introduced. Although they may not be specially designed for wireless sensor networks, these design concepts and experiences will benefit future implementations of positioning systems in wireless sensor networks.

1.5.1 Active Badge and Bat

1.5.1.1 Introduction Efficient location and coordination of staff in any large organization is a difficult and recurring problem. Hospitals, for example, may require up-to-date information about the location of staff and patients, particularly when medical emergencies arise. In an office building, a receptionist is usually responsible for determining the location of staff members; in some organizations, public address systems are provided to help a receptionist locate employees but, more frequently, a telephone is used to contact all the possible locations at which the required person might be found. These solutions can cause a great deal of irritation and disruption to other employees; a solution that provides direct location information is more desirable. See [38] for detail information.

1.5.1.2 Active Badge vs Pager The conventional solution used for personnel location is the *pager system*. In order to locate someone, a signal is sent out by a central facility that addresses a particular receiver unit (beeper) and produces an audible signal. In addition, it may display a number to which the called party should phone back.(Some systems allow a vocal message to be conveyed about the call-back number.) It is then up to the recipient to use the conventional telephone system to call back confirming the signal and to determine the required action. Although useful, in practice there are still circumstances where it is not ideal. For instance, if the called party does not reply, the controller has no idea whether they are in an area

where the signal does not penetrate, have been completely out of the area for some time, have been too busy to reply, or have misheard or misread the call-back number. Moreover, in the case where there are a number of people who could respond to a crisis situation, it is not known which one is the nearest to the crisis and therefore the most suitable to contact. An alternative approach is to *tag* a person and try to locate the tag. The main application of personal tags has been in the area of access control and logging [15]. In many high-security installations, card-key systems restrict access to various parts of the installation. If there are enough access-control zones, the same mechanism can also provide location information on a per-zone basis. For this kind of system, the information is positive and directly available to be acted upon. However, it is inappropriate for most organizations to use access-control techniques to derive location information because of the inconvenience experienced by personnel. There are additional problems arising from groups of people obtaining access to adjoining zones with only one card-key, which is a hard problem to solve.

1.5.1.3 An Active Badge Design A solution to the problem of automatically determining the location of an individual has been to design a tag in the form of an 'Active Badge' that emits a unique code for approximately a tenth of a second every 15 seconds (a beacon). These periodic signals are picked up by a network of sensors placed around the host building. A master station, also connected to the network, polls the sensors for badge 'sightings', processes the data, and then makes it available to clients that may display it in a useful visual form. The badge was designed in a package roughly 55x55x7mm and weighs a comfortable 40g. Pulse-width modulated infrared (IR) signals are used for signaling between the badge and sensor [31] mainly because: IR solid-state emitters and detectors can be made very small and very cheaply (unlike ultrasonic transducers); they can be made to operate with a 6m range, and the signals are reflected by partitions and therefore are not directional when used inside a small room. Moreover, the signals will not travel through walls, unlike radio signals that can penetrate the partitions found in office buildings. Infrared communication has been used in a number of commercial applications ranging from the remote control of domestic appliances to data backup links for programmable calculators and personal organizers [12]. More recently, IR has been used as the basis for wireless local area networks [27]. Because IR technology has already been exploited commercially, it is inexpensive and readily available for developing new applications such as the Active Badge. An active signaling unit consumes power; therefore, the signaling rate is an important design issue. Firstly, by only emitting a signal every 15 seconds, the mean current consumption can be very small with the result that 'badge-sized' batteries will last for about one year. Secondly, it is a requirement that several people in the same locality be detectable by the system. Because the signals have a duration of only one-tenth of a second, there is approximately a 2/150 chance that two signals will collide when two badges are placed in the same location. For a small number of people, there is a good probability they will all be detected. Even so, in order to improve this chance, the beacon oscillator has been deliberately designed around low-tolerance components. The components used for the beacon oscillator have a 10% tolerance rating; for two

badges to remain in synchronization for even a single 15-second beacon period, the components would have to be matched better than 1.4%. It is very likely that two badges, which at some instant may be synchronized, will have slightly differing frequencies and thus lose synchronization within a few beacon periods. In practice, synchronization has not been a problem. The Active Badge also incorporates a light-dependent component that, when dark, turns the badge off to conserve battery life. Reduced lighting also increases the period of the beacon signal to a time greater than 15 seconds. In ambient lighting conditions in a room, this effect only slightly modifies the period, but it is another factor that ensures synchronized badges will not stay synchronized very long. If the badge is placed in a drawer out of office hours, at weekends and during vacation, the effective lifetime of the batteries is increased by a factor of 4. Note that the more obvious solution of a manual switch was considered a bad idea as it was likely that a badge user would forget to turn it on. Other options for switching the device on included a tilt switch and an accelerometer, although the size limitation of a badge precluded using them in the initial experimental system. A disadvantage of an infrequent signal from the badge is that the location of a badge is only known, at best, to a 15-second time window. However, because in general a person tends to move relatively slowly in an office building, the information the Active Badge system provides is very accurate. An Active Badge signal is transmitted to a sensor through an optical path. This path may be found indirectly through a surface reflection, for example, from a wall. A badge must be worn on the outside of clothing, so an essential part of the badge case design was the clip allowing it to be easily attached to a shirt or a blouse. Most commonly, the badge was worn at the breast pocket position; however, some people preferred a belt or waist position. The belt position was not as good when the wearer was seated at a desk but typically the system still detected enough signals to locate the badge.

Active Badge location system developed at Olivetti Research Laboratory and now AT&T at Cambridge.

1.5.1.4 Bat System A successor of the Active Badge system is the Bat system [4], which consists of a collection of wireless transmitters, a matrix of receiver elements, and a central RF base station. The wireless transmitters, called bats, can be carried by a tagged object and/or attached to equipment. The sensor system measures the time of flight of the ultrasonic pulses emitted from a bat to receivers installed in known and fixed positions. It uses the time difference to estimate the position of each bat by trilateration. The RF base station coordinates the activity of bats by periodically broadcasting messages to them. Upon hearing a message, a bat sends out an ultrasonic pulse. A receiver that receives the initial RF signal from the base station determines the time interval between receipt of the RF signal and receipt of the corresponding ultrasonic signal. It then estimates its distance from the bat. These distances are sent to the computer, which performs data analysis. By collecting enough distance readings, it can determine the location of the bat within 3 cm of error in a three-dimensional space at 95% accuracy. This accuracy is quite enough for most location-aware services; however, the deployment cost is high.

1.5.2 Cricket

1.5.2.1 Introduction The design and deployment of a system for obtaining location and spatial information in an indoor environment is a challenging task for several reasons, including the preservation of user privacy, administration and management overheads, system scalability, and the harsh nature of indoor wireless channels. The degree of privacy offered by the system is an important deployment consideration, since people often value their privacy highly. The administrative overhead to manage and maintain the hardware and software infrastructure must be minimal because of the potentially large number (possibly several thousands in a building) of devices and networked services that would be part of the system, and the communication protocols must be able to scale to a high spatial density of devices. Finally, indoor environments often contain substantial amounts of metal and other such reflective materials that affect the propagation of radio frequency (RF) signals in nontrivial ways, causing severe multipath effects, dead-spots, noise, and interference. See [29] for more information about Cricket.

Cricket is a location-support system for in-building, mobile, location-dependent applications. It allows applications running on mobile and static nodes to learn their physical location by using listeners that hear and analyze information from beacons spread throughout the building. Cricket is the result of several design goals, including user privacy, decentralized administration, network heterogeneity, and low cost. Rather than explicitly tracking user location, Cricket helps devices learn where they are and lets them decide whom to advertise this information to; it does not rely on any centralized management or control and there is no explicit coordination between beacons; it provides information to devices regardless of their type of network connectivity; and each Cricket device is made from off-the-shelf components and costs less than U.S. $10.

By not tracking users and services, user privacy concerns are adequately met. We emphasize that Cricket is a *location-support* system, rather than a conventional *location tracking* system that tracks and stores location information for services and users in a centrally maintained database.

The design of Cricket was driven by the following specific goals:

- **User privacy:** Whenever a system for providing location information to clients has been deployed in the past, the issue of user privacy has arisen. This is because many previous systems were location tracking systems, where a database kept track of the locations of all the entities, including users in the system. To address this concern, we designed a location support system, which allows clients to learn their location without centralized tracking in order to construct location-specific queries for resources.

- **Decentralized administration:** Our goal is widespread building-wide deployment. We believe that it is not possible to deploy and administer a system in a scalable way when all control and management functions are centralized. Our design is decentralized - the ”owner” of a space in a building (e.g.,the

occupant of a room) configures and installs a location beacon that announces the identity of that space (some character string) and each beacon seamlessly integrates with the rest of the system. Location receiver hardware, called a listener, is attached to every device of interest to a user. Listeners use an inference algorithm to determine the space in which they are currently located by listening to beacon announcements. And there is no need to keep track of individual components within the system.

- **Network heterogeneity:** A wide variety of network technologies exist in most building environments. In our own laboratory, devices and users connected over 10/100 Mbps Ethernet, three different types of indoor wireless LANs, cellular digital packet data (CDPD), infrared, public telephone, and power line using X10 [1]. Independent of which technology they use to serve or gain access to information, many services and clients can benefit from learning their location in an automatic way, and we would like to accommodate them. In our design, we achieve this by decoupling the Cricket system from other data communication mechanisms.

- **Cost:** Achieving building-wide deployment requires cost effective components.We use commercial, off-the-shelf, inexpensive components in Cricket, setting and meeting the goal of less than U.S. $10 per location beacon and listener. Our design involves no custom hardware and is small enough to fit in one's palm.

- **Room-sized granularity:** Our goal is a system where spatial regions can be determined to within a few square feet, so as to distinguish portions of rooms. This requires the ability to demarcate and determine boundaries between regions corresponding to different beacons.

Cricket uses a combination of RF and ultrasound to provide a location-support service to users and applications. Wall- and ceiling-mounted beacons are spread through the building, publishing location information on an RF signal. With each RF advertisement, the beacon transmits a concurrent ultrasonic pulse. The listeners receive these RF and ultrasonic signals, correlate them to each other, and infer the space they are currently in. The beacons use a decentralized randomized transmission algorithm to minimize collisions and interference amongst each other. The listeners implement a decoding algorithm to overcome the effects of ultrasound multipath and RF interference.

1.5.2.2 System Architecture The only configuration required in Cricket is setting the string for a space that is disseminated by a beacon. The specific string is a function of the resource discovery protocol being used, and Cricket allows any one of several possibilities (in Section 5 we describe our implementation platform and integration with INS). Cricket also provides a way by which the owner of a room can securely set and change the space identifier that is sent in the advertisements. This is done by sending a special message over the same RF channel that is used for

the advertisements, after authenticating the user via a password. At this stage, we have chosen to allow this change only from within physical proximity of the room or location where the beacon is located. This makes the system somewhat more secure than if we allowed this to be done from afar. The boundaries between adjacent spaces can either be real, as in a wall separating two rooms, or virtual, as in a non-physical partition used to separate portions of a room. The precision of the system is determined by how well the listener can detect the boundary between two spaces, while the granularity of the system is the smallest possible size for a geographic space such that boundaries can be detected with a high degree of precision. A third metric, accuracy is used to calibrate individual beacons and listeners; it is the degree to which the distance from a beacon, estimated by a listener, matches the true distance. While our experiments show that the distance accuracy of our hardware is smaller than a few inches, what matters is the precision and granularity of the system. These depend on the algorithms and the placement of beacons across boundaries. Our goal is a system with a close-to-100of a few feet (a portion of a room).

1.5.2.3 Beacon positioning and configuration The positioning of a beacon within a room or space plays a nontrivial role in enabling listeners to make the correct choice of their location. For example, consider the positioning shown in Figure 2. Although the receiver is in Room A, the listener finds the beacon in Room B to be closer and will end up using the space identifier advertised by the latter. One way of overcoming this is to maintain a centralized repository of the physical locations of each beacon and provide this data to listeners. Systems like the Bat essentially use this type of approach, where the central controller knows where each wall- or ceiling-mounted device is located, but it suffers from two problems that make it unsuitable for us. First, user-privacy is compromised because a listener now needs to make active contact to learn where it is (observe that in Cricket, a listener is completely passive). Second, it requires a centrally managed service, which does not suit our autonomously managed environment particularly well. Fortunately, there is a simple engineering solution to this problem that preserves privacy and is decentralized. Whenever a beacon is placed to demarcate a physical or virtual boundary corresponding to a different space, it must be placed at a fixed distance away from the boundary demarcating the two spaces. Figure 3 shows an example of this in a setting with both real and virtual boundaries. Such placement ensures that a listener rarely makes a wrong choice, unless caught within a small distance (1 foot in our current implementation) from the boundary between two beacons advertising different spaces. In this case, it is often equally valid to pick either beacon as the closest.

1.5.3 RADAR:An In-Building RF-based User Location and Tracking System and Nibble

RADAR [8] is a radio-frequency (RF) based system for locating and tracking users inside buildings. RADAR operates by recording and processing signal strength information at multiple base stations positioned to provide overlapping coverage in

the area of interest. It combines empirical measurements with signal propagation modeling to determine user location and thereby enable location-aware services and applications.

RADAR complements the data networking capabilities of RF wireless LANs with accurate user location and tracking capabilities, thereby enhancing the value of such networks. RADAR uses signal strength information gathered at multiple receiver locations to triangulate the user's coordinates. Triangulation is done using both empirically-determined and theoretically-computed signal strength information.

Recently, several location systems have been proposed for wide-area cellular systems [35]. The technological alternatives for locating cellular telephones involve measuring the signal attenuation, the angle of arrival (AOA), and/ or the time difference of arrival (TDOA). While these systems have been found to be promising in outdoor environments, their effectiveness in indoor environments is limited by the multiple reflections suffered by the RF signal, and the inability of off-the-shelf and inexpensive hardware to provide fine-grain time synchronization.

Systems based on the Global Positioning System [28], while very useful outdoors, are ineffective indoors because buildings block GPS transmissions. RADAR differs from previous work in that it tackles the problem of user location and tracking on a widely available radio frequency based wireless network in an in-building environment. RF networks offer a significant advantage over IR (Infra Red) networks in terms of range, scalability, deployment, and maintenance. With speeds of up to 11 Mbps, these systems have gained rapid acceptance and are being widely deployed in offices, schools, homes, etc.

RADAR has been developed by a Microsoft Research group. It is a building-wide tracking system based on the IEEE 802.11 WaveLAN wireless networking technology [8]. RADAR measures, at the base station, the signal strength and signal-to-noise ratio of signals that wireless devices send, then it uses this data to compute the 2D position within a building. Microsoft has developed two RADAR implementations: one using scene analysis and the other using lateration. The RADAR approach offers two advantages: It requires only a few base stations, and it uses the same infrastructure that provides the buildings general-purpose wireless networking. Likewise, RADAR suffers two disadvantages. First, the object it is tracking must support a wireless LAN, which may be impractical on small or power-constrained devices. Second, generalizing RADAR to multi-floored buildings or three dimensions presents a nontrivial problem. RADARs scene-analysis implementation can place objects to within about 3 meters of their actual position with 50 percent probability, while the signal-strength lateration implementation has 4.3-meter accuracy at the same probability level. Although the scene-analysis version provides greater accuracy, significant changes in the environment, such as moving metal file cabinets or large groups of people congregating in rooms or hallways, may necessitate reconstructing the predefined signal-strength database or creating an entirely new database.

Several commercial companies such as WhereNet [2] and Pinpoint [3] sell wireless asset-tracking packages, which are similar in form to RADAR. Pinpoints 3D-iD

performs indoor position tracking using proprietary base station and tag hardware to measure radio time of flight. Pinpoints system achieves 1- to 3-meter accuracy and, by virtue of being a commercial product, offers easier deployment and administration than many research systems. The 3D-iD system suffers the disadvantage that each antenna has a narrow cone of influence, which can make ubiquitous deployment prohibitively expensive. Thus, 3D-iD best suits large indoor space settings such as hospitals or warehouses. It has difficulty interoperating with the 802.11 wireless networking infrastructure because of radio spectrum collision in the unregulated Industrial, Scientific, and Medical (ISM) band.

The Nibble also adopts the IEEE 802.11 infrastructure for positioning purpose. Nibble uses the probability-based approach in Subsection 1.2.2.2. It relies on a fusion service to infer the location of an object from measured signal strengths. Data are characterized probabilistically and input into the fusion service. The output of the fusion service is a probability distribution over a random variable that represents some context.

1.5.4 CSIE/NCTU Indoor Tour Guide

The prototype indoor tour guide system has been developed at the Department of Computer Science and Information Engineering, National Chiao Tung University (CSIE/NCTU), Taiwan. The hardware platforms of this project include several Compaq iPAQ PDAs and laptops. Each mobile station is equipped with a Lucent Orinoco Gold wireless card. Signal strengths are used for indoor positioning. The probability-based pattern-matching algorithm in Subsection 1.2.2.2 is used.

There is a manager or control center responsible for monitoring each users movements, configuring the system, and planning logical areas and events. The location server takes care of the location discovery job and the service server is in charge of message delivery. The database can record users profiles; the gateway can conduct location-based access control to the Internet. One of the innovations in this project is that an event-driven messaging system has been designed. A short message can be delivered to a user when he enters or leaves a logical area. The event-driven message can also be triggered by a combination of time, location, and property of location (such as who is in the location and when the location is reserved for meetings). A user can set up a message and a corresponding event to trigger the delivery of the message. The manager will check the event list periodically and initiate messages, when necessary, with the service server. Messages can be unicast or broadcast. The expectation is that streaming multimedia can be delivered in the next stage. The system can also be applied to support a smart library. Another innovation is to provide location-based access control. In certain rooms, such as classrooms and meeting rooms, users may be prohibited from accessing certain sensitive Web pages. These rules can be organized through the manager and set up at the gateway.

1.6 CONCLUSION

In this chapter, some fundamental techniques in positioning and location tracking have been discussed and several experimental systems reviewed. Location information may enable new types of services. Accuracy and deployment costs are two factors that may contradict each other, but both are important factors for the success of location-based services.

References

1. X-10 home page: http://www.x10.com/homepage.htm.

2. Widata home page: http://www.widata.com/.

3. Pinpoint home page: http://www.rftechnologies.com/pinpoint/.

4. Mike Addlesee, Rupert Curwen, Steve Hodges, Joe Newman, Pete Steggles, Andy Ward, and Andy Hopper. Implementing a sentient computing system. *Computer*, 34(8):50–56, 2001.

5. Localization James Aspnes. On the computational complexity of sensor network.

6. S. Atiya and G.Hager. Real-time vision-based robot localization. pages 785–800, 1993. IEEE Trans. Robot. Automat. 9 (6).

7. P. Bahl, A. Balachandran, and V. Padmanabhan. Enhancements to the radar user location and tracking system, 2000.

8. Paramvir Bahl and Venkata N. Padmanabhan. RADAR: An in-building RF-based user location and tracking system. In *INFOCOM (2)*, pages 775–784, 2000.

9. P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. *ACM/Kluwer Wireless Networks*, 7(6):609–616, 2001. 3rd int. Workshop on Discrete Algorithms and methods for mobile computing and communications, 1999, 48-55.

10. N. Bulusu, J. Heidemann, and D. Estrin. Gps-less low cost outdoor localization for very small devices, 2000.

11. Srdan Capkun, Maher Hamdi, and Jean-Pierre Hubaux. GPS-free positioning in mobile ad-hoc networks. In *HICSS*, 2001.

12. S. L. Harper, Worsley R. L., and B. A. Stephens. An infrared link for low-cost calculators and printers. In *Hewlett-Packard Journal*, October 1987.

13. Andy Harter, Andy Hopper, Pete Steggles, Andy Ward, and Paul Webster. The anatomy of a context-aware application. In *Mobile Computing and Networking*, pages 59–68, 1999.

14. J. Heidemann and N. Bulusu. Using geospatial information in sensor networks, 2001.

15. P. Hewkin. Smart tags the distributed memory revolution. In *IEEE Review*, June 1989.

16. Jeffrey Hightower and Gaetano Borriella. Location systems for ubiquitous computing. *IEEE Computer*, 34(8):57–66, 2001.

17. Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David E. Culler, and Kristofer S. J. Pister. System architecture directions for networked sensors. In *Architectural Support for Programming Languages and Operating Systems*, pages 93–104, 2000.

18. Brad Karp and H.T. Kung. Gpsr: Greedy perimeter stateless routing for wireless networks. In *Proc. of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, 2000.

19. Y. Ko and N. Vaidya. Geocasting in mobile ad hoc networks: Location-based multicast algorithms, 1998.

20. Y. Ko and N. Vaidya. GeoTORA: A protocol for geocasting in mobile ad hoc networks. Technical Report 00-010, Dept. of Computer Science, Texas A&M University, March 2000.

21. Koen Langendoen and Niels Reijers. Distributed localization in wireless sensor networks: a quantitative comparison. *Comput. Networks*, 43(4):499–518, 2003.

22. J. Leonard and H.Durrant-Whyte. Mobile robot localization by tracking geometric beacons. pages 376–382, 1991. IEEE Trans. Robot. Automat. 7 (3).

23. D. Li, K. Wong, Y. Hu, and A. Sayeed. Detection, classification, and tracking of targets, 2002.

24. J. Li, J. Jannotti, D. De Couto, D. Karger, and R. Morris. A scalable location service for geographic ad-hoc routing. In *Proceedings of the 6th ACM International Conference on Mobile Computing and Networking (MobiCom '00)*, pages 120–130, 2000.

25. Julio C. Navas and Tomasz Imielinski. GeoCast – geographic addressing and routing. In *Mobile Computing and Networking*, pages 66–76, 1997.

26. Dragos Niculescu and Badri Nath. Ad hoc positioning system (APS). In *GLOBECOM (1)*, pages 2926–2931, 2001.

27. A. Paepcke, R.D. Crawford, R. Jamp, C. A. Freeman, and F.J. Lee. Chipnet: An optical network of terminals andworkstations. In *Elsevier Science Publishers B. V. (North-Holland), Computer and ISDN Systems 15*, pages 203–215, 1988.

28. P.Eng and P.Mirsa. Special issue on global positioning system. In *Proceedings of the IEEE*, volume 87, pages 3–15, January 1999.

29. Nissanka B. Priyantha, Anit Chakraborty, and Hari Balakrishnan. The cricket location-support system. In *Mobile Computing and Networking*, pages 32–43, 2000.

30. T. Roos, P. Myllymaki, H. Tirri, P. Misikangas, and J. Sievanen. A probabilistic approach to wlan user location estimation. In *Int. J. Wireless Inf. Networks*, 2002.

31. Satellite, Cable, and TV IC Handbook. In *Plessey Semiconductors*, pages 64,67,124, 1988.

32. C. Savarese, J. Rabay, and K. Langendoen. Robust positioning algorithms for distributed ad-hoc wireless sensor networks usenix technical annual conference, 2002.

33. A. Savvides, H. Park, and M. Srivastava. The bits and flops of the n-hop multilateration primitive for node localization problems, 2002.

34. Andreas Savvides, Chih-Chieh Han, and Mani B. Strivastava. Dynamic fine-grained localization in ad-hoc networks of sensors. In *Mobile Computing and Networking*, pages 166–179, 2001.

35. S. Tekinay. Wireless geolocation systems and services. In *Special Issue of the IEEE Communications Magazine*, April 1998.

36. R. Tins, L. Navarro-Serment, and C. Paredis. Fault tolerant localization of teams of distributed robots, 2001.

37. Y.-C. Tseng, S.-P. Kuo, H.-W. Lee, and C.-F. Huang. Location tracking in a wireless sensor network by mobile agents and its data fusion strategies. In *Int. Workshop Inf. Process. Sensor Networks (IPSN)*, 2003.

38. Roy Want, Andy Hopper, Veronica Falcão, and Jonathan Gibbons. The active badge location system. Technical Report 92.1, Olivetti Research Ltd. (ORL), 24a Trumpington Street, Cambridge CB2 1QA, 1992.

39. M. Youssef, A. Agrawala, and U. Shankar. Wlan location determination via clustering and probability distributions. In *IEEE PerCom*, 2003.