Infrastructureless Location Aware Configuration for Sensor Networks[*]

ISI-TR-2004-589

Xi Wang Fabio Silva John Heidemann
{xiw, fabio, johnh}@isi.edu

Abstract

In large sensor networks nodes must self-configure their communication, location, and other characteristics. GPS and similar systems determine location today, but they require substantial infrastructure in the environment or on sensor nodes to locate nodes in a physical coordinate system. For many applications, *logical* location—the relationship of nodes with each other and their environment—can be more important than physical location. For example, distance along a road and presence of intersections may be more relevant than Euclidean coordinates for applications that track or guide drivers. In this paper we present a novel algorithm, *deployment order*, for logical location determination. Deployment order exploits node deployment patterns and simple user interactions to define logical topologies in a completely distributed manner. With minimal user interaction it can establish arbitrarily complex logical topologies. We illustrate the algorithm through the "follow-me" application, which is an easy-to-deploy sensornet guidance system suitable for use in office buildings as well as inhospitable environments (underground, in damaged buildings, etc.). Finally, we demonstrate how the addition of landmarks allows the conversion from logical locations to approximate physical locations.

1 Introduction

Sensor networks use numerous small, inexpensive nodes that can sense, compute, and communicate with each other to interact with the physical world. Sensors must be small and inexpensive to make it reasonable so that there can be many of them; many are needed to allow them to be physically present throughout the environment.

This combination of small, inexpensive, numerous devices deployed in the physical world leads to the focus of this paper. Deployment and *configuration* of these devices pose major challenges. Unlike carefully engineered traditional networks, the large number of devices implies that deployment of a sensor network is not necessarily well planned. Nodes can be deployed rapidly in an ad hoc fashion, forming large scale sensor networks with relatively short life times. In addition, sensor network configuration may require consideration of aspects of the physical environment. For these reasons, automatic configuration of a sensor network is both essential and challenging.

Nodes in sensor networks interact closely with their surrounding environment, and one of the most important parameters in many sensor network applications is location. Two different kinds of location are often required: both *physical* location, coordinates in some frame of reference, and *logical* location, often application-specific knowledge such as which sensors are adjacent to others, or which are in a room or hallway. In some sensor applications, logical location can be more important and more difficult to determine than physical location.

In this paper we describe a novel approach to determine the logical location of sensors. We use the *follow-me* application to illustrate "walkable connectivity", one type of logical location. We describe *deployment order*, a novel approach to capture this logical location. A unique characteristic of deployment order is that it requires no fixed infrastructure such as GPS receivers or other localization-specific hardware, thus it is applicable to very small, inexpensive nodes. We have implemented a sensor-network-based guidance system for visitors, and we studied our configuration approach on the real system.

2 Problem Description

To motivate the challenges of deployment order we first introduce the *follow-me* application. We then give an overview of our approach to configuration.

Although we describe a specific application and a configuration approach for logical location, both generalize to other scenarios. Follow-me represents a class of applica-

[*]In this work John Heidemann and Xi Wang are partially supported through the NSF Division of Civil and Mechanical Systems, grant number E01-CMS-0112665. John Heidemann and Fabio Silva are also partially supported by USC/CSULB METRANS 2003–04 grant SURE-SE.

tions where sensors are deployed to assist navigation. Other examples include marking paths in buildings damaged by earthquake or fire, or underground exploration. Sensor nodes can guide people and sense the environmental hazards at the same time.

The need for logical location also extends beyond just these applications. In many applications, both indoors and outside, logical position information is not immediately apparent from physical node deployment.

2.1 The Follow-Me Application

The problem of automatic configuration with logical location information can be illustrated by a sensor network based visitor guidance system.

For a visitor stepping into an office building for the first time, navigating unknown places can be difficult and unpleasant. While signs may guide the way, and computer kiosks may provide room numbers and maps, neither provides active assistance to visitors as they move through a building.

The follow-me application is an active visitor guidance system designed to address this problem. Sensor nodes are deployed around a building, on walls, one at each office doorway. Nodes blink their lights to indicate a path, guiding a visitor with a "breadcrumb trail" to the destination.

2.2 Our Approach

To guide visitors, the follow-me application needs to automatically find a path between a source node and a destination node, and show this path in to a visitor.

Node Deployment Node deployment in the follow-me application is based on two general guidelines: There should be one node at each office doorway, and the distance between two adjacent nodes should not be too large. The later means we need to place additional nodes along hallways with few doors, such that visitor can follow lights easily. We show several possible node deployment examples later (see Figure 7).

Path Finding and Logical Location The follow-me application must guide visitors along appropriate paths. While network routing algorithms specialize in path finding, they are not directly applicable for guiding humans, who are constrained by physical walls and prefer to follow adjacent nodes. Traditional routing algorithms select the shortest path based on radio connectivity, selecting paths through physical walls and skipping physically intermediate nodes when possible. Even a strictly geographic routing algorithm will cut corners and pass through walls if it shortens the physical path.

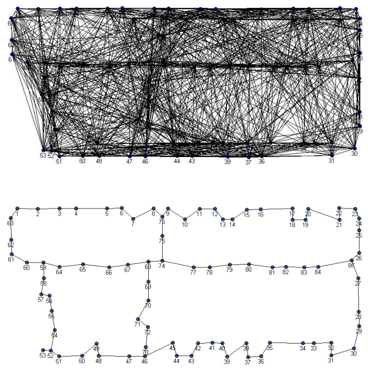


Figure 1: Comparison between radio connectivity graph (top) and logical topology (bottom).

Thus the main technical challenge in follow-me is determining the *logical topology* that connects nodes as a human would walk, as opposed to the radio or physical topologies.

Figure 1 compares radio and logical topologies for the same follow-me deployment in Figure 7(a).

The location of a node is represented by its neighbors in the logical topology, we call this the *logical location* of a node.

2.3 Configuring Logical Location

Each node in the follow-me application needs to be configured properly with its logical location, which is a set of physical neighbors. Identifying correct physical neighbors for each node is an important configuration problem.

We would like nodes to configure themselves automatically. With localization techniques, it is possible to estimate logical locations from physical coordinates; we review a number of systems in Section 7. However existing techniques do not directly apply to applications that require logical location for several reasons. First, logical location is defined by human constraints such as walls and doors; these constrains are not easily visible to typical localization techniques based on RF or ultrasound. Second, follow-me requires building-like topologies: long, linear segments, parallel hallways with moderate density nodes. These topologies are especially unfriendly for distance-triangulation based localization techniques. Finally, we desire a system that is easy to deploy and has low cost. Pre-deployment of substantial infrastructure or extensive measurements (for example, as with RADAR [1]) greatly raise the cost of a system and preclude ad hoc deployments.

For these reasons we chose to develop a new method for

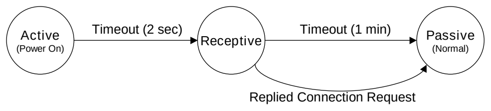


Figure 2: Initial state diagram for deployment order.

logical location configuration. In Section 3 we will describe *deployment order*, our algorithm that captures logical topology. It is present when a network is first configured, allowing construction of complex topologies with minimal human interaction.

2.4 Interacting with Users

A common and effective approach of designing sensor network applications is to keep sensor nodes simple, and rely on the collaborative behavior of the whole network to achieve complex functions. Unlike systems with a keyboard and a screen, simple components such as LEDs and buttons are more frequently used, and sensor nodes are spatially distributed in the target environment. The user interface part of the follow-me application shares the same idea.

There are three kinds of user interactions involved: visitors need to tell network their destinations; network needs to show path to visitors; and during deployment network administrators need to interact with individual sensor nodes for configuration.

3 Deployment Order for Logical Location Configuration

In many instances sensor nodes are deployed sequentially, perhaps being dropped one by one by a single person or vehicle. Deployment order takes advantage of this information by assuming that when two nodes are deployed (switched on) one after the other within a short time, we can assume that they are closest neighbors to each other. Links between these closest neighbors can create a path corresponding to connectivity of an individual walking through a building. If nodes can detect and remember this path, it can be used later to guide visitors. Some other mechanisms are needed to handle non-linear topologies such as *intersections*. One method is to manually interact with sensor nodes to add and remove links. We will discuss both linear paths and intersections below.

3.1 Linear Paths

To create a linear path, a newly deployed node communicates with previously deployed nodes to determine which one was deployed immediately prior to the deployment of

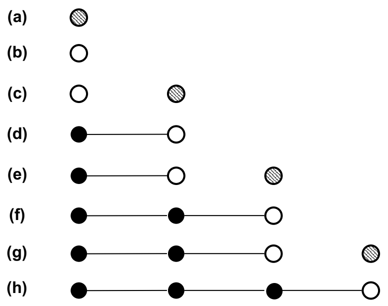


Figure 3: Linear topology example for deployment order handling intersections. States - gray: active, white: receptive, black: passive.

itself. If such a node is found, these two nodes should link to each other. We realize this with the following simple state machine on each node (see Figure 2):

Active This is the state after a node is switched on. Nodes in this state send out connection request packets to look for neighbors.

Receptive Nodes in this state will reply to connection request packets (from active nodes). A reply establishes a link between the two nodes.

Passive Nodes in this state will not be involved in link operations. This is the state for normal operation.

Nodes begin in active state where they find their previous neighbor, wait in receptive state to pick up the next neighbor, and then transition to passive state. Figure 3 is an example of a linear path. After the first node is switched on (a), it won't find any neighbor and will go to receptive state (b). When the second node is switched on (c) it begins in active state and will search for neighbors. The first node (currently receptive) will reply, establishing a link between these two nodes (c). The first node will move to passive state after creating the link, and the second will go to receptive state (d). Similarly, the third node will link to the second node, and so on (e)-(h).

This simple state machine is sufficient to create linear topologies. The user does not need to know the details of the state machine, merely that the last node in the line is "hot" (receptive) and will connect to the next node that is turned on. To provide confirmation about the network state we provide both visual and audio feedback about the network as it is deployed. Nodes in active state have all LEDs lit up. When an active node detects its neighbor, the active node transitions to just a red LED to indicate it is now receptive, and the neighbor beeps and transitions to a green LED (indicating it is now passive). For linear deployments the person deploying the network can be completely oblivious

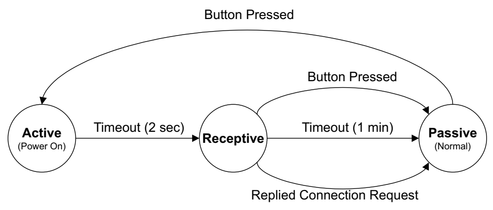


Figure 4: Revised state diagram for deployment order handling intersections.

to the state machine, simply listening for beeps to confirm that each step of the configuration is complete.

3.2 Intersection Handling

Linear paths are automatically configured while nodes are being deployed without external infrastructure or any explicit user involvement. For richer topologies, we must consider *intersections* where nodes have more than two neighbors. Assuming the majority of links belong to linear paths, we can still use the basic process for linear path on most nodes. We add a very simple interaction process to handle intersections.

The key observation on handling intersections is that if we give users a little bit of control over the state machine, they can then connect nodes to make arbitrary topologies.

In our implementation, we use a button on sensor node to toggle node states. When a node is in passive state, pressing the button will bring the node to active state. When a node is in receptive state, pressing the button will bring the node to passive state. The updated state diagram is shown in Figure 4.

With the ability to change state, we can add arbitrary connections by making one node active and another node receptive. An example for intersections is shown in Figure 5. In this figure, the logical topology consists of two linear segments joined at a shared central node. To create this topology, we first deploy one linear segment with the basic deployment order procedure described earlier. Then we deploy the second segment until it meets the first segment. By pressing a button, the central node of the first segment is put into active state. It will link with the newly deployed node of the second segment and become an intersection node. After this step, the central node is in receptive state and we can continue the deployment of the second segment until the whole topology is created. We can see there is only one button press needed for an intersection.

3.3 Fault Handling

A sensor network must tolerate node failures. Ultimately, arbitrary failures can be handled by patching in new sensors using the same mechanisms for creating intersections. However, we also would like a mechanism to handle single node failures.

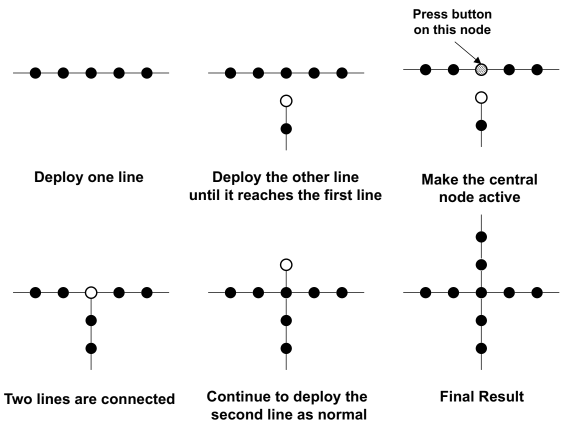


Figure 5: Intersection handling for deployment order.

Nodes monitor their neighbors and detect failure by a repeated lack of response. Since wireless networks are unreliable, using a simple threshold may misjudge some cases. Studying better methods for detecting failures is left for future research.

Once a failed neighbor is identified, the detecting node will skip this neighbor and link to the neighbor's neighbors directly. This is done by broadcasting a “link fix” packet containing the ID of the failed neighbor. Neighbors of the failed node will respond with their own IDs, allowing the detecting node to establish links and fill in any gaps. This process assumes that radio range is at least twice the distance between neighbors, a typical configuration since radios in sensor networks easily reach 20m indoors or more and node placement is typically 3-5m apart.

4 Design and Implementation of Follow-Me

With the deployment order method, logical location information is available to applications. In this section more topics about the design and implementation of the follow-me application are presented. Important design issues include path finding algorithm and user interaction processes. Most implementation issues are related to interactions with lower layers.

4.1 Path Finding

Given the logical topology it is relatively easy to find a path. We can use a simple minimum-distance routing algorithm over the logical topology to determine the best path between two points for a visitor. Our current implementation uses flooding to find forward paths and gradient style routing for reverse paths. This routing combination is very similar to directed diffusion [14]. Other routing algorithms could also be used, provided they operate on logical topology.

When a visitor arrives at the lobby and selects a destination from a touch screen, the network finds the path as described above, flooding and establishing previous-hop gradients. The destination node gathers routes and selects the best one based on the desired metric. A good metric would be physical distance traveled. Our current implementation assumes all nodes are equidistant and so hop count is equivalent to physical distance. Using the method to infer physical position described later in Section 6, one could compute approximate physical distances to improve routing of visitors.

While we use routing in the logical topology for follow-me, a general routing service is available to many tasks. For example, we monitor our network from a central point using this same routing algorithm. In this case our routing algorithm uses radio connectivity rather than logical topology.

4.2 User Interactions with the Network

There are two types of users in the follow-me application: visitors and network administrators. We first discuss user interactions for visitors and then cover network administration issues.

4.2.1 Network/Visitor Interactions

Communication between visitors and network are two way: visitors need to tell network about their destinations, and network needs to show path to visitors. In our design a touch screen is used for visitors to choose their destinations, and synchronized blinking patterns across the network are used for showing paths to visitors.

For network to visitor communication, blinking patterns should create a visual effect of moving light dots or lines, conveying both path and direction information to visitors in an intuitive way. Alternatively, synchronized beeping can also be used.

To produce blinking patterns, timing parameters including phase and interval are used. We use a command packet to carry these parameters and trigger the sequence. The source node at entrance sends out the command packet at the beginning of a blinking sequence. There is no packet transmission needed afterwards.

Before we trigger a blinking sequence using timing parameters, nodes need to be time-synchronized. Several sensor-network specific time synchronization protocols have recently been proposed [7, 9]. However the timing precision requirement of this application is not very high, so we implement very simple time synchronization by using the command packet to define a time base without correcting for clock drift or transmission delay. We do exploit MAC-level support for time synchronization as described in Section 4.3.2.

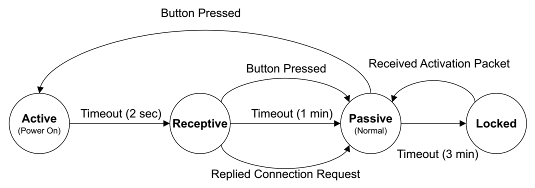


Figure 6: State diagram with node locking

These are the underlying mechanisms that support network/visitor interactions. An interesting step further is to guide multiple visitors at the same time. One possible solution is to show several paths simultaneously using different colors or blinking patterns.

4.2.2 Interactions During Configuration and Node Locking

User interaction about network configuration is usually the first type of user interactions activated for a sensor network. During the configuration process of the follow-me application, network administrators switch on nodes to create linear paths and press buttons to create intersections for the deployment order method.

After a network is configured, inadvertent configuration changes could occur if someone accidentally pressed buttons on sensor nodes during normal operation. To prevent this situation, buttons need to be locked after the node configuration process is complete. On the other hand network administrators may still want to change configurations periodically for network maintenance. We designed a locking mechanism that utilizes a *key* node to fulfill these goals.

After a node is in passive state for three minutes, the node switches to locked state. It will not respond to button presses. To re-enable buttons on the node, the key node is used. This node will send out activation packets periodically. Nodes receiving this packet will unlock their buttons. A person who wishes to reconfigure the follow-me system can simply switch on a key node and carry it with him/her. The state diagram with this update is shown in Figure 6.

4.2.3 Configuring Destinations

Deployment order method can configure nodes' logical locations. To guide visitors directly to someone's office, we still need to match nodes' logical locations with offices, which is also a configuration problem. This configuration information can be represented by a table of node ID associated with room numbers and people's names.

We propose an interactive way to simplify the configuration process. A user who wishes to configure a node can approach the node with a PDA capable of communicating with the sensor network. By pressing a button on the node, the node's ID will be shown on the PDA's screen and information about the office can be entered.

4.3 Implementation and Lower Layer Interactions

The follow-me application is implemented on MICA2 sensor nodes with TinyOS operating system. We are in the process of completing the deployment shown in Figure 7(a) at ISI. As of April 2004, our current deployment is smaller, with eight nodes covering one long hallway at half the desired density, and with two nodes with labeled buttons substituting for the touch-screen display. Although this deployment is linear, the current implementation supports configuration of arbitrary topologies and locking.

Nodes in sensor networks usually belong to deeply embedded systems, where algorithms interact closely with lower level software and underlying hardware. In the following paragraphs we discuss issues and solutions for our implementation of follow-me.

4.3.1 Energy Conservation

Sensor nodes in the follow-me application run on battery power. As we want to put sensor nodes beside office doorways, power outlets are rarely available. Adding many power outlets in a building is generally a rather expensive task, easily exceeding the cost of sensor nodes. Thus even in an indoor environment we would like the follow-me application to be completely wireless without power cords.

Our goal is to have the system run on a single set of batteries for months. We use S-MAC [17] as a low-energy MAC protocol to allow radios and potentially the node CPU sleep most of the time. Currently CPU deep-sleep is not supported by S-MAC, thus there is an extra current drain during sleep period. With current S-MAC at 10% duty cycle, each follow-me node runs for 14 days on two AAA alkaline batteries. In our tests MICA2DOT nodes worked till the battery voltage dropped to 2.2v, which corrsponds to battery capactivy of about 1200mAh according to manufactures's manual. We plan to enable deep sleep mode and use lower duty cycles to further extend battery life.

4.3.2 Time Synchronization

The user interface of the follow-me application needs time synchronization to produce blinking patterns. During implementation, we encountered latency problems caused by the MAC layer. There are two kinds of latencies from S-MAC: latency caused by collision avoidance and latency caused by sleep cycles. These delays can severely affect time synchronization as they are rather long and higher layers do not have information about them.

Although there are time synchronization mechanisms dealing with various delays, letting MAC layer inform higher layers about these delays is a much more efficient way. At our request, Wei Ye added an optional timestamp that marks when a packet is transmitted by the physical layer. This mechanism is similar to that used by Ganeriwal et al. [9].

4.3.3 Latency and Route Caching

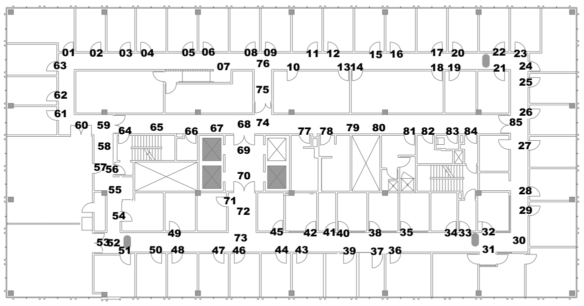
Packet forward latency increases when the MAC layer has sleep cycles. As the shortest MAC active period is limited by timing and energy consumption constrains, sleep period must be longer for lower duty cycle modes. This means energy saving goals may conflict with network reaction time goals.

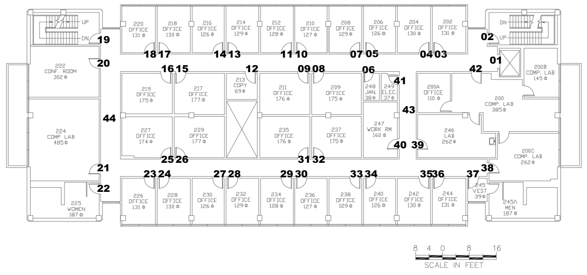
While follow-me does not have strict real-time requirements, it needs to be "fast enough" that users don't notice excessive delay. Ideally, routes should be discovered in a second or two. When routes are based on walking distance, they often will span dozens of hops and latencies exceed this limit.

To meet this latency goal while conserving energy we plan to cache paths in our application. Each node will have a table that associates source-destination pairs with their position in the path from source to destination. If a source-destination pair is found in the table, we only need to do a multihop broadcast to all nodes, and each node can derive information about whether it is in the path and what are the timing parameters for blinking. Packets only need to go through all radio hops, which is usually much less than the hops in the logical topology. Thus the latency can be significantly reduced.

5 Evaluation of Deployment Order

Manual configuration is needed for creating intersections with the deployment order method. We can consider the deployment order method an efficient method if only a small portion of nodes require manual configuration. Therefore we use the number of manual configurations as a performance metric to evaluate the deployment order method in


(a) ISI


(b) SAL

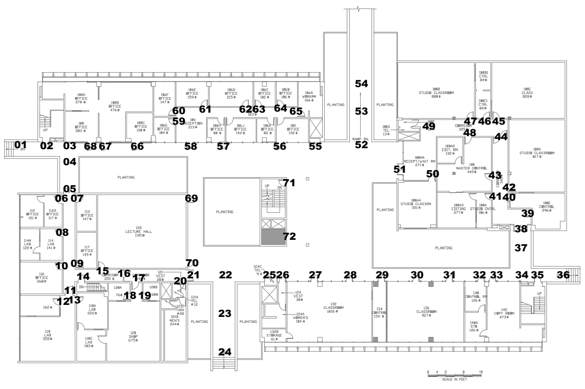

(c) OHE

Figure 7: Node deployment scheme for three building maps

this section.

We used maps of three buildings shown in Figure 7 to simulate and evaluate the deployment order method. The numbers in these maps indicate where nodes are placed.

These maps are chosen based on topology diversity and availability. We want to use real building maps with different types of topologies for the evaluation. ISI is the 11th floor of the Information Science Institute, SAL and OHE are from two building in the University of Southern California.

Table 1 shows the number of manual configurations needed for creating logical topologies of these buildings.

The effectiveness of the deployment order method depends on topology. Relatively simple topologies of ISI and SAL lead to small numbers of manual configurations. With many branches, the more complex topology of OHE leads to more manual configurations, however the majority of nodes can still configure their logical locations automatically. While the cost of deployment order grows in complex topologies, simple topologies require very little effort, and even the 17 manual configurations required for the OHE deployment require relatively little time compared to placing 72 nodes.

Building	Nodes	Manual Configurations	Ratio
ISI	85	7	8.2%
SAL	44	3	6.8%
OHE	72	17	24%

Table 1: Number of manual configurations

6 Physical Location Estimation

The primary goal of deployment order is to provide logical location information ("walkable connectivity"), because such information is impossible to get from traditional localization techniques. However, applications such as network visualization benefit from the addition of physical coordinates. In this section we show how the addition of *landmark* nodes with known positions enables us to estimate the physical coordinates from their logical locations for these applications.

6.1 Using landmarks to infer physical location

Landmarks are a few nodes with known locations. These known locations are combined with logical locations to give physical location estimations. On the assumption that the network topology consists of mostly linear segments with reasonably homogeneous node density, we infer the locations of nodes placed logically between landmarks by assuming they are evenly spaced. This assumption is appropriate for the buildings that we consider as potential follow-me deployments provided landmarks are chosen at the building corners.

Inspired by previous work in network topology visualization, we adapt a spring-embedder system [8] to approximate physical locations between landmarks. In such algorithms, simulated forces between nodes drive them to positions where attractive and repulsive forces are balanced, in effect the system converges to a state with lowest spring-embedder energy.

We hold landmarks as fixed and allow other nodes to move using the spring-embedder model. Parameters are set such that forces between nodes are mostly attractive. Combined with landmark nodes, this setting results in nearly straight lines as are typical in office buildings.

We expect this algorithm to run as an off-line processing task on a host system because it requires high processing power and runs infrequently. To connect the location estimation task with sensor nodes, the host system needs to collect logical location information from individual sensor nodes first. Our general-purpose routing service based on flooding and gradient style routing can be used for this task. Next, coordinates of landmark nodes need to be sent to the host system. Depending on the application, this information can either come from outside the network (perhaps encoded from the touch-screen host), or it could be deployed within the system, perhaps using a PDA as described in Section 4.2.3. The estimation algorithm then combines these two pieces of information and runs the spring-embedder model to derive physical coordinate estimations. Finally, derived coordinates may need to be sent back to sensor nodes for some applications. Again, gradient style routing (through the reverse path) can be used here as a simple solution.

Currently our location estimation algorithm runs on a PC and is not directly connected to the follow-me network. Since our current deployment is rather small, we simulate the results presented below to evaluate this approach.

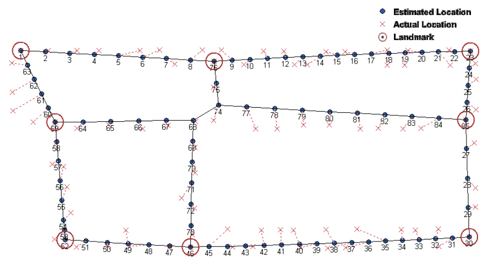
6.2 Accuracy of inferred physical locations

We evaluate the accuracy of this algorithm for inferring physical location with our three sample buildings.

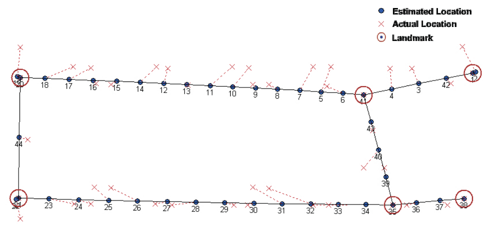
Figure 8 shows the result of the location estimation algorithm, corresponding to maps in Figure 7 with two possible anchor sets for OHE. In these figures, circled dots are landmark nodes with known locations. Dots are estimated positions. Crosses indicate true physical locations as references. Figure 9 shows location error distributions of these examples, which are the results of comparing nodes' true physical locations with locations obtained from the estimation algorithm.

As expected, we can see that the quality of result depends on the building topology and landmark placement. For these three maps, ISI and SAL have more accurate results than OHE. Placements for ISI and SAL are quite good in general, with 80% of estimated node locations within 1m of their true locations. As our earlier evaluation on number of manual configurations, the large number of branches in the OHE topology also makes the location estimation algorithm less accurate.

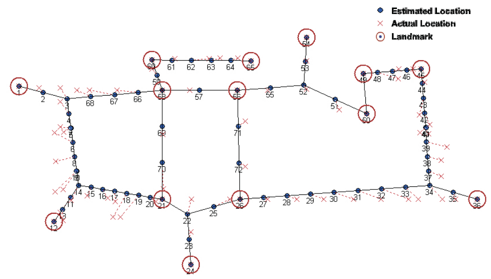
Accuracy could be improved by adding more landmark points. In Figure 8(d), we add five more landmarks to the OHE topology. This change results in noticeable im-



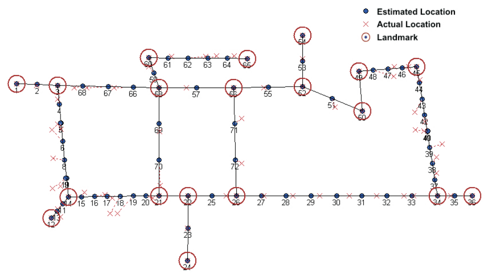
(a) ISI



(b) SAL



(c) OHE (14 landmarks)



(d) OHE (19 landmarks)

Figure 8: Location estimation of three buildings

provement on precision: In Figure 8(d) about 80% of nodes are now within two meters away from their true locations, while in Figure 8(c) they can be as far as three meters away. Hence this location estimation approach has some flexibility in balancing precision and the number of landmarks.

Applications that require highly accurate node locations may justify the addition of dedicated node hardware or in-

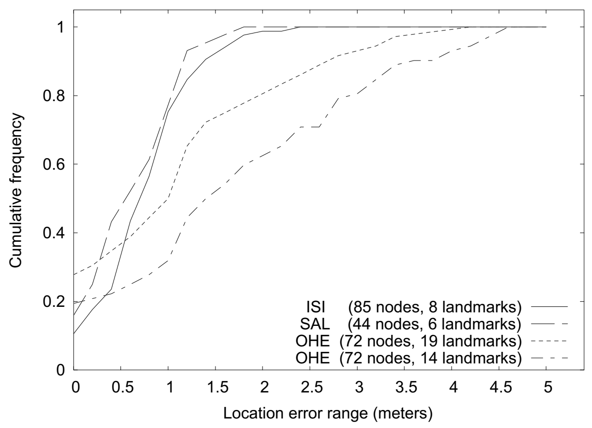


Figure 9: Error distribution of location estimation results

frastructure to allow extremely accurate node placement. However, we are encouraged that the combination of logical location from deployment order with a few landmark nodes allows approximation of physical location with reasonable accuracy and no additional hardware or infrastructure. This approach should be appropriate for applications that do not require very precise locations.

7 Related Work

There are three areas that are closely related to this paper. First, this paper belongs to the general area of location aware computing. Second, other researchers have related deployment and configuration for different applications. Finally, physical coordinate based localization is a different but connected area.

Location aware computing is an active topic in both sensor networks and pervasive computing [2, 12]. It is closely related to context-aware computing [4].

For location aware applications, location information is essential. Physical coordinates are rarely used directly as a source of location information because they are raw data without apparent meaning. Logical location [11] and symbolic location [13] are similar concepts and they can be a good source of location information. The deployment order method presented in this paper is designed to provide logical location directly and efficiently to sensor network applications.

Other researchers have used physical deployment to infer information about system configuration. Researchers at Berkeley have previous used a linear deployment order to infer node locations at a demonstration at Twentynine Palms in 2001 [15]. An unmanned aerial vehicle dropped six sensor nodes (Berkeley Rene motes) in a line. Sensors inferred their positions through recomputed knowledge of drop order assuming an initial fixed distance of deployment. With deployment order we instead allow an arbitrary number of identical nodes to determine their relative locations, and we identify how to build non-linear topologies with simple user interaction.

In this paper our approach is different from localization techniques as logical location is used instead of physical location, yet localization is still a closely related topic. Here we have a limited review about related localization techniques [13] based on the requirements of the follow-me application.

GPS [6] or other time-of-flight localization techniques [10] need support from specialized hardware, while some other techniques require little additional hardware, such as connectivity [5, 3] or RF signal strength based techniques.

Some form of infrastructure is frequently required for localization. Infrastructures can be GPS satellites, a group of sensor nodes with known locations [5, 3] or detailed maps of environmental features, such as multi-source RF signal strength information used in RADAR [1].

Finally, outdoor and indoor environments can make a significant difference for localization. The interior of a building (the scale here is larger than a single room) can be challenging. In such an environment, GPS signal is blocked by walls, multi-path effects are common for local radio signals, and acoustic waves are constrained to individual rooms.

Because of these difficulties, logical location information from deployment order is especially valuable for indoor applications.

The “resurrecting duckling” model is an analogy about security mechanisms for small wireless devices [16]. In the paper authors suggested using direct physical contact as a method of authentication. Their method provides a very simple solution compared to typical public-key-based authentication or third-party servers. In some ways our use of deployment order is similar, also using physical proximity during deployment to infer logical connectivity, with both avoiding the need for centralized infrastructure. The application domains are quite different, however.

8 Conclusion

We developed and evaluated deployment order, a new method for configuration of logical locations of nodes. To demonstrate the deployment order method, we implemented and deployed follow-me, an application that can guide users around buildings. We also evaluated the addition of landmarks to compute estimated physical node locations from logical locations.

These techniques are not appropriate for all sensor network applications. We expect applications that require exact physical location to warrant deployment and use of specialized hardware or infrastructure such as per-node GPS receivers, ultrasound, or other approaches. However, deployment order fills two roles not possible by these approaches: a very small, light-weight approach for approximate location when simpler software and minimal hardware is required; and the consideration of logical connectivity such as "walkable connectivity" required by applications such as follow-me.

References

[1] Paramvir Bahl and Venkata N. Padmanabhan. RADAR: An in-building RF-based user location and tracking system. In *Proceedings of the IEEE Infocom*, pages 775–784, Tel Aviv, Israel, March 2000. IEEE.

[2] H.P.W. Beadle, B. Harper, G.Q. Maguire, Jr., and J. Judge. Location aware mobile computing. In *Proceedings of the IEEE/IEE International Conference on Telecommunications*, pages 1319– 1324, Melbourne, Australia, April 1997. IEEE.

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

[4] Guanling Chen and David Kotz. A survey of context-aware mobile computing research. Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College, November 2000.

[5] Lance Doherty, Kristofer S. J. Pister, and Laurent El Ghaoui. Convex position estimation in wireless sensor networks. In *Proceedings of the IEEE Infocom*, pages 1655–1663, Anchorage, Alaska, USA, April 2001. IEEE.

[6] G. Dommety and Raj Jain. Potential networking applications of global positioning systems (GPS). Technical Report TR-24, CS Dept., The Ohio State University, August 1996.

[7] Jeremy Elson, Lewis Girod, and Deborah Estrin. Fine-grained network time synchronization using reference broadcasts. In *Proceedings of the Fifth USENIX Symposium on Operating Systems Design and Implementation*, pages 147 – 163, Boston, MA, USA, December 2002. USENIX.

[8] T.M.J. Fruchterman and E.M. Reingold. Graph drawing by force-directed placement. *Software—Practice and Experience*, 21(11):1129–1164, November 1991.

[9] Saurabh Ganeriwal, Ram Kumar, and Mani B. Srivastava. Timing-sync protocol for sensor networks. In *Proceedings of the First ACM Conference on Embedded Networked Sensor Systems*, pages 138 – 149, Los Angeles, California, USA, November 2003. ACM.

[10] Lewis Girod, Vladimir Bychkovskiy, Jeremy Elson, and Deborah Estrin. Locating tiny sensors in time and space: A case study. *Proceedings of the IEEE International Conference on Computer Design*, September 2002. Invited paper.

[11] John Heidemann and Nirupama Bulusu. Using geospatial information in sensor networks. In *Proceedings of the Workshop on Intersections between Geospatial Information and Information Technology*, Arlington, VA, USA, October 2001. National Research Council.

[12] John Heidemann and Dhaval Shah. Location-aware scheduling with minimal infrastructure. In *USENIX Conference Proceedings*, pages 131–138, San Diego, CA, June 2000. USENIX.

[13] Jeffrey Hightower and Gaetano Borriello. Location systems for ubiquitous computing. *IEEE Computer*, 34(8):57–66, August 2001.

[14] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proceedings of the ACM International Conference on Mobile Computing and Networking*, pages 56–67, Boston, MA, USA, August 2000. ACM.

[15] Kris Pister et al. Twentynine Palms fixed/mobile experiment. web page `http://robotics.eecs.berkeley.edu/~pister/29Palms0103/`, March 2001.

[16] Frank Stajano and Ross Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. In *Security Protocols, 7th International Workshop Proceedings*, pages 172–194, 1999.

[17] Wei Ye, John Heidemann, and Deborah Estrin. An energy-efficient MAC protocol for wireless sensor networks. In *Proceedings of the IEEE Infocom*, pages 1567–1576, New York, NY, USA, June 2002. USC/Information Sciences Institute, IEEE.