

DOLPHIN: An Autonomous Indoor Positioning System in Ubiquitous Computing Environment

Yasuhiro FUKUJU*, Masateru MINAMI*, Hiroyuki MORIKAWA**, and Tomonori AOYAMA*

* Graduate School of Information Science and Technology, The University of Tokyo

** Graduate School of Frontier Sciences, The University of Tokyo

{y-fukuju, minami, mori, aoyama}@mlab.t.u-tokyo.ac.jp

Abstract

Determining physical location of indoor objects is one of the key issues in development of context-aware applications in ubiquitous computing environment. This is mainly because context information obtained from sensor network is meaningful only when the physical location of context information source is determined. Recently, several indoor location information systems, such as Active Bat[1] and Cricket[2], have been developed for precise indoor object localization. However, to provide accurate physical location tracking in large-scale space, those systems requires a lot of manual configuration for all the ultrasonic sensor nodes. To reduce configuration costs, we developed a new indoor positioning system called DOLPHIN. The DOLPHIN system consists of distributed wireless sensor nodes which are capable of sending and receiving RF and ultrasonic signals. These nodes are attached to various indoor objects. And using a novel distributed positioning algorithm in the nodes, DOLPHIN enable autonomous positioning of the objects with minimal manual configuration. This paper describes the design and implementation of the DOLPHIN system, and evaluates basic performance through several experiments in indoor environment.

1. Introduction

In ubiquitous computing environment, physical location of various indoor objects is one of the key information to provide context-aware applications. In outdoor environment, we can easily obtain precise location information from GPS (Global Positioning System). However, in contrast to outdoor environment, indoor environment usually requires different kind of positioning system because usually GPS signal is not available. To obtain location information in indoor environment, several indoor positioning systems have been developed.

SpotOn[3] and RADAR[4] are RF-based indoor positioning system that use RF signal, such as wireless LAN and Bluetooth, to locate indoor objects. These systems measure distance between transmitter and receiver based on RSSI (Received Signal Strength Information) of

RF signal, and compute location of object using the information. However, it is difficult to measure precise distance using received signal strength, and the estimated position usually varies by a few meters. Of course, it is possible to utilize UWB (Ultra Wide Band) technology to measure precise range based on TDOA (time-difference-of-arrival), but it usually requires complicated hardware.

Active Bat [1] and Cricket [2] use ultrasonic pulse TDOA to enable high precision 3-dimensional positioning in indoor environment. Although, ultrasonic positioning system requires additional hardware to send and receive ultrasonic pulse, it can determine 3D position of indoor objects with an accuracy of a few mm to a few cm.

However, both RF-based and ultrasonic-based systems usually require many manual pre-configuration of the locations of reference stations in order to determine objects position precisely. This indicates that setup and management costs would be unacceptable if we apply them to large scale environment such as an office building.

From this point of view, we have developed a new ultrasonic positioning system called DOLPHIN (Distributed Object Locating System for Physical-space Internetworking). In the DOLPHIN system, location of objects is automatically determined in a distributed manner using only few manually configured references. Although, current version of the DOLPHIN system employs ultrasound-based approach, it is possible to apply positioning algorithm to RF-based system. In our system, all objects in the system have capability of sending and receiving ultrasonic and radio signals to measure distance between two objects with only a few manually preconfigured objects as reference stations. All other locations of objects are gradually determined based on a recursive positioning algorithm.

This paper describes the positioning algorithm and prototype implementation of the DOLPHIN system as well as its experimental evaluation. The paper is organized as follows. In section 2, we briefly summarize system structure of the DOLPHIN system. Next, in section 3, we introduce a distributed recursive positioning algorithm to determine object position using a few reference stations. In section 4, we report the experimental evaluation of

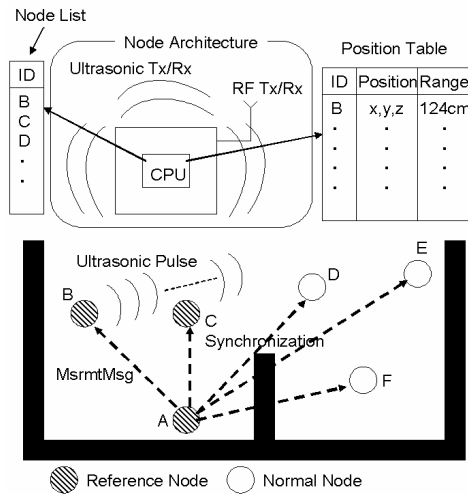


Fig. 1: The DOLPHIN System

implemented DOLPHIN system, and summarize the paper in section 5.

2. The DOLPHIN System

Figure 1 shows the system architecture of DOLPHIN system. The system consists of nodes that have RF and ultrasound transmission/reception function and one-chip CPU for calculating location of the nodes. The RF function is used for time synchronization and message exchange among nodes. The CPU has a pulse counter, and it is used to measure the TDOA of ultrasonic pulse.

The basic positioning principle of the DOLPHIN system is similar to the Active Bat system and Cricket. For example, reference node A in the figure 1 transmits RF signal, then all other nodes start internal pulse counter when they receive the RF signal. The RF signal contains predetermined position of node A. Meanwhile, node A transmits ultrasonic pulse to other nodes. When a node receives the ultrasonic pulse, it stops internal counter and compute its distance to node A based on the speed of sound. If the node collects 3 or more entries of the distance information from nodes that positions have been determined, it can compute 3D position using GPS-like positioning algorithm.

Different from Active Bat or Cricket System, the DOLPHIN system requires only a few pre-configured reference nodes for locating all other nodes in the system. The idea of DOLPHIN positioning is based on hop-by-hop locating mechanism. For example, in figure 1, node D can determine its position by receiving ultrasound pulse from the reference nodes A, B, and C. However, node E and F cannot receive all ultrasonic pulses from reference nodes because of obstacles (e.g. wall, etc.). Here, if the position

of node D is determined, and node E receives ultrasonic pulse from node D, then node E can compute its position by using distances from node B,C and D. And if the location of node D and E is determined, node F can compute its position using node C, D and E. In this way, all nodes in the DOLPHIN system can be located.

There are two main advantages in this mechanism. First, the system requires only a few (minimum three) nodes to determine all position of nodes. Second, nodes can determine their position even if the nodes cannot receive ultrasound from reference nodes directly. We designed this mechanism based on a distributed algorithm, and the next section describes the details of the algorithm.

3. Positioning Algorithm

3.1 Basic Algorithm

In the DOLPHIN system, there are two types of nodes: a *reference node*, which is a fixed node located to previously measured position, and a *normal node* which location is determined by the algorithm. Each node has a unique ID for RF communication, a node list for node selection, and a position table for position calculation.

In our positioning algorithm, three types of messages are exchanged among nodes. These messages are transmitted via RF signal. An *ID notification message (IDMsg)* is used for notifying a node ID to other nodes. A *measurement message (MsrmtMsg)* triggers distance measurement among nodes. This message contains node ID of a certain node which should transmit ultrasonic pulse and used for time synchronization in the DOLPHIN system. A *location notification message (LocMsg)* contains ID and location of the node which transmits the message.

By exchanging these messages, all nodes in the system play three different roles: one *master node*, one *transmitter node*, and many *receiver nodes*. A master node selects one

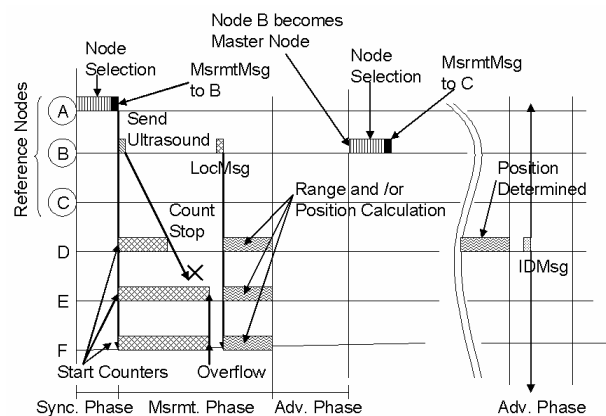


Fig. 2: Positioning Algorithm

transmitter node based on the node list, then transmits measurement message with a certain ID. When a node receives the message and its ID matches the transmitted ID, it becomes transmitter node and subsequently generates ultrasonic pulse. At the same time, all other nodes become receiver nodes and start their internal counter. If a receiver node detects an ultrasonic pulse, it stops internal counter and computes the distance between the node and transmitter node. Note that, in our algorithm, only nodes whose location is determined can become transmitter node.

Consider an example shown in figure 1. In this example, node A, B, and C are reference node, and node D, E, and F are normal nodes. Here, we assume that Node A, B, and C have node lists $[B, C]$, $[A, C]$, and $[A, B]$ respectively. And we also assume that node E and node F could not receive ultrasonic pulse from Node A because of obstacle such as a wall.

Now we consider that node A acts as a master node. Figure 2 shows the timing chart of our positioning algorithm. First, node A chooses one node randomly from its node list $[B, C]$. If node B is chosen, node A transmits *MsrmtMsg* including ID of node B. On receiving the message, node B becomes transmitter node and generates ultrasonic pulse. At the same time, other nodes C, D, E, and F become receiver node and start their internal counter (*synchronization phase*). If receiver nodes detect ultrasound, they stop their internal counter and calculate the distance from node B. After several ms (this depends on the time taken by the overflow of internal counter), node B sends *LocMsg* to notify its position to receiver nodes. Receiver nodes that could detect ultrasound store location of node B and distance to node B in their position table (*measurement phase*). After that, all nodes wait for listening *IDMsg* for several ms (*advertisement phase*). If any node which could determine position based on three or more distance, it advertises ID in this phase. And the ID is added to node list in every node. In the above example, because node D, E, F cannot determine their position, no *IDMsg* is sent in this phase. The sequence of above phases is one cycle of the positioning algorithm in the DOLPHIN system.

In the next cycle, node B which acts as receiver node in previous cycle becomes master node. And the positioning algorithm proceeds in the same way. After three or more cycles of positioning, node D determines its position based on measured distance from node A, B, and C. At that time, node D sends *IDMsg* in the advertisement phase. All other nodes that received *IDMsg* from node D add ID of node D to their node list, and node D is recognized as a candidate of master node.

Once after node D becomes master node, node E and node F can measure distance from node D. Then, node E

determines its position and advertises *IDMsg*. Finally, based on node C, D and E, node F can determine its position. In this way, we can locate all nodes in the DOLPHIN system.

3.2 Recovery from Failures

In the DOLPHIN system, we have to prepare for two types of failures, node failure and recognition failure, to continuously execute the above mentioned positioning algorithm. The node failure is that the node suddenly stops because of unpredictable accident, and the recognition failure is that the *IDMsg* transmitted from the node capable of master node does not reach other node because of bad communication channel or message collision.

To recover from node and recognition failures, each node in the system has a recovery timer and advertisement timer. The recovery timer is set when nodes receive *MsrmtMsg*, and expires if there has been no *MsrmtMsg* for a certain period (e.g. 15 seconds). If the recovery timer expires, a node in the system is chosen to become master node randomly, and the positioning algorithm continues.

If a node capable of master node does not receive *MsrmtMsg* from other nodes within a certain period (e.g. 10 seconds), the advertisement timer in the node expires. Thus, that means the node is not recognized as a node capable of master node by other nodes. In this case, the node retransmits *IDMsg* at advertisement phase in each positioning cycle. Note that to avoid *IDMsg* collision at advertisement phase, the node sends *IDMsg* at a certain probability which determined by the number of nodes in the node list.

3.3 Bootstrap

At the initial state of the system, node list in each node is empty. Therefore, to start positioning, each node must collect node IDs in bootstrap phase. The recovery timer and advertisement timer are used for this purpose.

Figure 3 shows bootstrap algorithm of the DOLPHIN system. After turning on the system, an advertisement timer in a reference node expires, and the reference node sends *IDMsg*. On receiving the *IDMsg*, every node in the system recognizes the reference node and registers the node ID in their node list. After that, reference nodes that received *IDMsg* start recovery timer. When a recovery timer in a reference node expires, the reference node becomes master node. In this way, every node collects node IDs of reference nodes during bootstrap phase, and the system enters the positioning cycle described in the previous section.

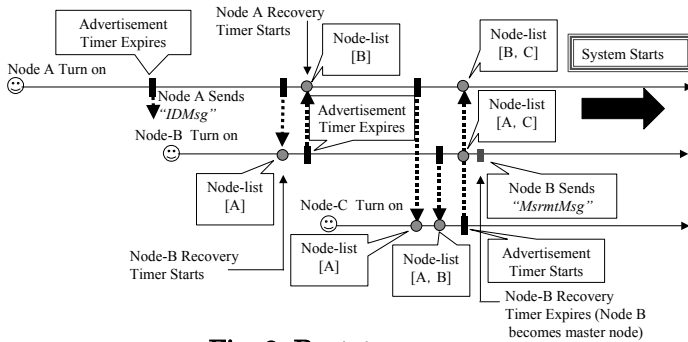


Fig. 3: Bootstrap

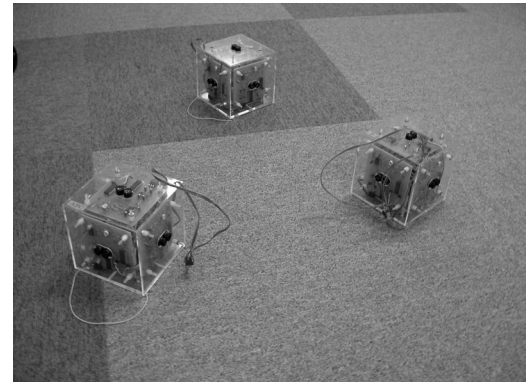


Fig. 4: Implemented Nodes

4. Implementation and Experiments

Figure 4 shows a prototype implementation of the DOLPHIN system. In our implementation, we build cubic nodes (each edge of the cube is 10cm long). We put ultrasonic transmitter and receiver (MA40S4S/R, Murata) on top and 4 sides of the cube, and insert a control circuit and battery pack inside the cube. We utilized STR232 (Nomura Engineering) as the RF communication device and Microchip PIC16F877 for implementing the positioning algorithm.

We placed seven nodes as shown in figure 5, and computed the average and the variance of the measured position of each normal node (node D-G) for 1000 cycles. In figure 5, the circled shot patterns denote measured positions, and x_{ave} , y_{ave} , σ^2_x , σ^2_y denote the average position of x , y and the variance of x , y , respectively. In this experiment, node D, E, F and G determine their position based on nodes A-C, nodes B-E, nodes C-E, and nodes D-F respectively. In the experimental results, because nodes A, B, and C have precise position, node D which refers those nodes can determine its position precisely (positioning error less than 5cm). However, positioning error becomes 10-15cm at nodes E-F that measure their position based on node D. This is because the positioning error at node D affects position determination of nodes E-F that determine their position based on node D. Although, this error propagation problem is inherently unavoidable in the DOLPHIN system, we expect that it is possible to minimize positioning error by placing reference nodes at appropriate location.

5. Summary

In this paper, we described the design and implementation of the DOLPHIN system. A distributed algorithm to determine location of indoor objects was introduced, and experimental evaluation showed that the

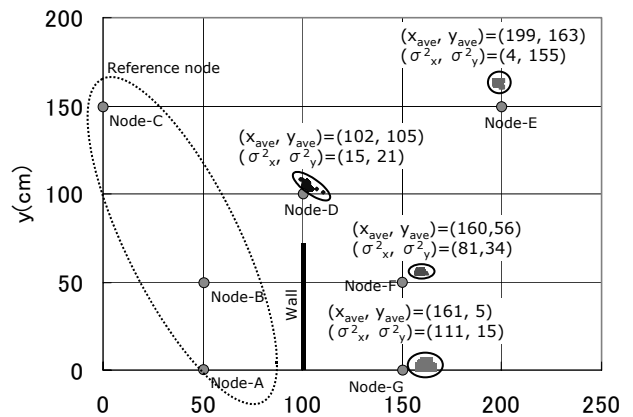


Fig. 5: Experimental Result

designed positioning works well with an accuracy around 15cm. However, the accuracy will be degraded when hop count between reference and normal nodes increases. Solving this error propagation problem will be our future work and we are determined to improve the accuracy of the DOLPHIN system.

References

- [1] A.Harter, A.Hopper, P.Steggles, A.Ward, and P.Webster, "The Anatomy of a Context-aware Application," MOBICOM1999, August 1999.
- [2] N.Priyantha, A.Miu, H.Balakrishnan, and S.Teller, "The Cricket Compass for Context-aware Mobile Applications," MOBICOM2001, July 2001.
- [3] J.Hightower, C.Vakili, G.Borriello, and R.Want, "Design and Calibration of the SpotON Ad-Hoc Location Sensing System," <http://portolano.cs.washington.edu/projects/spoton/>, August 2001.
- [4] P.Bahl and V.N.Padmanabhan, "RADAR: An In-Building RF-based User Location and Tracking System," IEEE INFOCOM2000, Vol.2, March 2000.