Implementation-based Approach for Designing Practical Sensor Network Systems

Masateru Minami†, Shunsuke Saruwatari‡, Takuya Kashima‡, Takashi Morito†,
Hiroyuki Morikawa‡, and Tomonori Aoyama‡

†Shibaura Institute of Technology (3-9-14 Shibaura, Minato-ku, Tokyo, Japan)
‡The University of Tokyo (7-3-1 Hongo, Bunkyo-ku, Tokyo, Japan)

Abstract – *Wireless sensor network technologies are expected to be a key technology to support various innovative applications in the future ubiquitous computing environment.*

So far, the main stream of sensor network research has been focused on simulation-based development of battery-aware communication protocols. However, when we apply sensor network technologies to practical applications such as environmental monitoring or factory automation, we will find a lot of practical problems that are not considered in simulation-based approach. We believe that finding and solving such practical problems are quite important for deploying sensor network technologies beyond laboratory use.

From this point of view, we have been trying to find such problems through various implementations of sensor network system for several years. This paper describes our opinions of designing sensor network system and introduces our approaches for developing practical systems.

Keywords: Sensor Network, Software Development Kit, Battery-awareness, Localization

1 Introduction

Integration of physical information in computer networks is expected to enhance the potential of computer network applications beyond traditional e-mail and web applications in the future ubiquitous computing environment. For example, location information will enable context-aware applications that provide appropriate services depending on user's situations and global-scale physical information will provide new knowledge to the natural sciences. Usually, such physical information will be obtained from sensor networks embedded in physical space.

Generally, the term "sensor network" includes wide variety of technologies, but wireless sensor network is attracting great deal of attention because of its flexibility. Usually, a wireless sensor network is considered as a distributed system which consists of huge number of tiny sensor nodes. Each node has a microcontroller, wireless communication device, and sensors. In particular, to disseminate sensors in various environments, sensor nodes are assumed to be battery-powered, and this defines various features of wireless sensor network; each sensor node has limited resources and communication protocols must be lightweight and battery-aware.

To tackle these technical challenges, there have been a lot of researches on wireless sensor networks including software, hardware and communication technologies[1]. Among these researches, designing battery-aware communication protocols has been the main stream of research on wireless sensor networks for recent years. However, in a practical sense, only developing battery-aware communication protocols is insufficient to develop workable sensor network systems and deploy it in actual environment. In addition, almost all researches are done based on computer simulations, and this blinds actual problems that should be solved in wireless sensor network systems. Needless to say, we do not mean simulation-based approach is meaningless, but feedbacks from both implementation and management of practical system are also important to design practical sensor network systems.

In contrast to this trend, recently, various wireless sensor network platforms, such as MICA Mote[2] and SmartIts[3], have been developed. In particular, MICA Mote is the pioneer work in implementing wireless sensor network system, and several practical problems are found out through implementing applications on it. However, since MICA Mote is designed as a general purpose system, it implements quite fundamental functionalities to construct wireless sensor network system.

On the other hand, to support practical applications, there still remain a lot of problems that should be solved through various implantations and experimentations. Verification of previously proposed communication protocols and development of viable localization scheme are examples of such problems that we can not neglect.

From this point of view, we have been implemented and evaluated various hardware and software systems to inclusively tackle to practical problems in wireless sensor network for several years. Through our work, we concluded that a flexible testbed for implementing various communication protocols and applications, battery-less technology for supplying power to each sensor node, and precise localization mechanism are essentials to design practical sensor network systems.

In the following sections, we introduce our developed technologies that are our solutions to the previous three charenges.

2 Flexible Sensor Network Testbed

2.1 Background

Wireless sensor networks have been assumed being used in outdoors such as battlefield and environment monitoring. Many research groups have proposed a number of routing protocols and MAC (media access control) protocols, for battery powered and resource limited sensor nodes. The researches have studied using computer simulation. An attractive concept of wireless sensor networks had much effect on a lot of people, and many applications of wireless sensor networks start to be considered. To create new applications and to validate proposed protocols in practical use of the applications, a hardware testbed for wireless sensor networks is much awaited.

Toward these demands, several hardware platforms, like MICA mote or Smart-Its, have been developed. These platforms realized miniaturization and low-cost by transacting communication tasks and application tasks in one CPU. However, implementing the tasks to one CPU can't fulfill the diverse demands of application and communication mechanisms for wireless sensor networks. For example, an application developer is restricted to implement application if he uses time sensitive MAC protocols in one CPU architecture.

From these points of view, we design and implement a software and hardware framework for wireless sensor networks "PAVENET" which supports to construct various applications[5]. PAVENET includes U^3 that is a hardware module for development, U^3 SDK that is software development kit for U^3, and basenode software that supports development of application and operates with personal computers and PDAs.

U^3 has two CPU oppose to MICA or Smart-Its: one CPU is for application, the other CPU is for communication. This 2 CPU architecture provides us some merits in following points. First, it gives us to develop applications and communication protocols easily. Typically, an application developer and a communication protocol developer are two different developers. Therefore not considering overhead of application and communication's tasks each other is important to develop software easily. Next, it gives us to evaluate communication protocols, such as MAC protocols, easily. Since communication protocols and application are separate in hardware, we can purely evaluate the communication protocol's characteristics like power consumptions. Finally, it helps us to consider oncoming development in hardware. If we examine low power consumption and miniaturization of wireless sensor nodes, we have to implement a part of functions in hardware. Especially, since implementing media access control protocols in hardware expects reducing much energy consumption, dividing communication and application tasks to 2 CPU is effective.

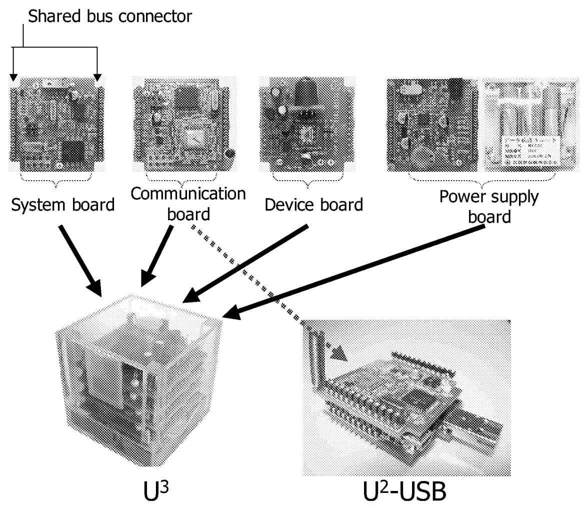

Figure 1: Hardware organization

2.2 Architecture

We implemented software and hardware framework for wireless sensor networks PAVENET, which provides support to construct various applications. PAVENET includes U^3 that is a hardware module for development, U^3 SDK that is software development kit for U^3, and basenode software that supports development of application and operates with personal computers and PDAs.

Figure 1 shows each boards, and U^3 and U^2-USB that are constructed by four boards.

U^3 is a $50mm \times 50mm \times 50mm$ box that contains four function boards which are a power board, a system board, a communication board, and a device board. And four boards are power supply board, system board, communication board, and device board. The boards are connected by a 2.54 mm pitch shared bus connector with each other.

U^2-USB consists of communication board and I^2C -USB conversion board. It is a communication interface to control the wireless sensor nodes from personal computers.

The following describes the details of each board.

The power supply board has three AAA 700 mAh nickel metal-hydride batteries and external DC input for charge. Furthermore, it supplies 3.0 V to other boards, gives information about battery life, and output current to the shared bus connector. System board can know remaining amount of battery life and energy consumption through the shared bus connector.

The communication board consists of an RF Monolithics 315Mhz transceiver TR3001, a Microchip 8-bit MCU PIC18F452 that runs at 20MHz, and a helical antenna. PIC18F452 has 8-bit registers, 1.5KB data memory, 16KB program memory, 256 bytes EEPROM, and controls TR3001, and processes communication software of U^3

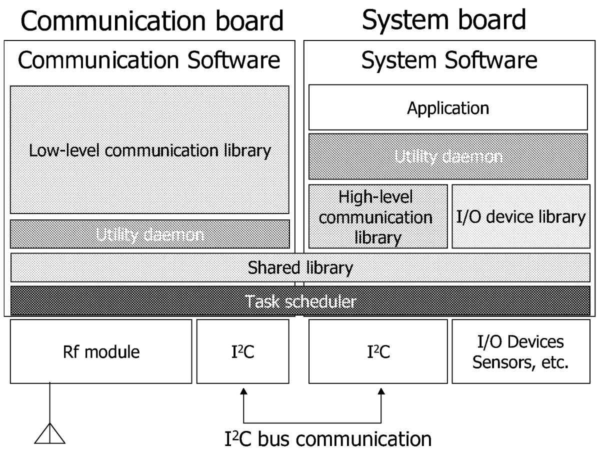

Figure 2: U^3 SDK

SDK. The communication board provides I^{2}C interface to the shared bus connector.

The system board consists of the same Microchip PIC18F452 as communication board and IrDA 1.0. PIC18F452 runs at 10 MHz, and processes system software of U^3 SDK.

The device board can equip various sensors or actuators, which are controlled by the system board. The device board provides some interfaces to the shared bus connector. The interfaces includes voltage that presents information such as temperature, a port that presents 1 bit data, a port that controls the device, and I^{2}C interface.

Currently, we have implemented sensor board that has a motion sensor, a temperature sensor, and an illuminance sensor, for test.

U^3 SDK is software development kit, and supports system software and communication software which are processed by system board and communication board, respectively. We use HI-TECH Software PICC18 compiler for the development of U^3 SDK. U^3 SDK is designed strongly concerning network layering, and supports to realize various users' demands.

Figure 2 shows the structure of U^3 SDK. Both of the system software and the communication software have the same task scheduler and shared library. The task scheduler has lightweight multithreading architecture, and supports hard realtime transaction. Shared library consists of various APIs that are for task operation such as add_task, trigger_task, resume_task, etc., and for debugging such as exit_u3.

The system software consists of high-level communication library, I/O device library, and utility daemon. As the components are triggered by interruption and the transaction are designed to complete immediately, we can keep a large percentage of CPU state idle, and achieve low power consumption.

High-level communication library transacts data aggregation and adhoc routing, etc. It is said that packets shall route data centrally and application specifically in wireless sensor networks[4]. Users can expand functions, such as routing, by describing transaction in event handlers like on_net_recv. U^3 SDK also provides APIs, which enable us to gain independence from various protocols. Hence, we can replace network protocols by trial and error. The APIs include set_net_opt or get_net_opt which are used when setting destination addresses or getting source addresses.

I/O device library provides a structure that abstracts interfaces such as I^{2}C , UART, ADC, etc., by open/read/write functions.

The communication software consists of low-level communication library and utility daemon.

The low-level communication library transacts communication functions in physical layer and MAC layer, which will be implemented in LSI in the future. In PAVENET, these functions are implemented by software with consideration of prospective implementation in hardware because of flexibility of software. Especially, the low-level communication library supports for a part of adhoc routing functions such as source routing, flooding, etc. in simple network layer. These simple adhoc routing functions can be realized in low memory consumption and simple scheme. Hence, they can be implemented by hardware. To transact a part of adhoc routing in communication software reduces loads of system software, and it will accomplish prospective low power consumption and miniaturization.

Utility daemons give us an interface to set protocol parameter, and to record communication log by providing control interface layer. Control interface layer works with low-level communication library.

The basenode software runs on PC or PDA to utilize wireless sensor networks. It consists of command line utility, protocol translation gateway, and basenode library.

Command line utility can be used from command line, and includes pvnload which loads program to U^3, pvnget/pvnset which can get/set parameters of U^3, and pvnping which is used for confirming sensor nodes existence or used for measurement of packet delivery time.

Protocol translation gateway works on PC that equips U^2-USB , and it allows users to access to sensor networks through the Internet.

The basenode library provides APIs which are arranged from functions of command line utility, and it is used when user develops an application for wireless sensor network without command line utility.

2.3 Applications

We are developing a sensor network application called ANTH (ANtennary Things) using PAVENET[6].

In ubiquitous computing environment, communication and computation functions are embedded in every object around us, and this enable us to synthesize various services by combining these objects.

Many service coordination frameworks, such as UPnP, Jini, and Bluetooth, have been proposed until now. These frameworks are useful to construct conventional services or

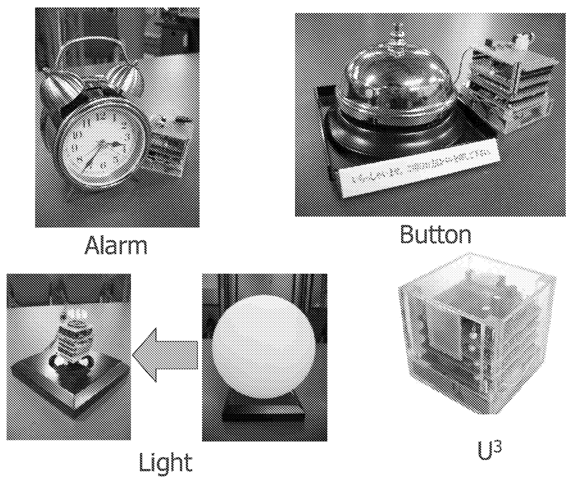

Figure 3: Prototype Implementation

static services, because they are designed for configuring devices automatically or replacing cable with wireless. These technologies remove annoying entwining cables and complex device configuration from us. However, they never provide us creative environment that enables us to construct various services.

In view of this, we are developing real-space programming framework called ANTH (ANtennary THings) for ubiquitous computing environment. ANTH aims to construct programmable real-space that enables us to create our desired services by ourselves. ANTH provides a chip that has three characteristic functions: a user interface function that controls connecting one device and another device, a communication function that constructs a network infrastructure for device cooperation, and a computation function which drives devices and processes applications. The chip is called ANTH chip, and an object that equips an ANTH chip is called ANTH object. We assume, in the future, ANTH chips are embedded to all everyday objects around us, such as alarm clocks, lights, walls, and so on.

Figure 3 shows 4 implemented ANTH objects: a light, an alarm clock, a button, and a U^3 which is wireless sensor node. The light and the alarm clock bell work as SNs, the alarm clock timer, the motion sensor of U^3 , and the button work as ENs, and laptop acts as a *controller*. We have tested the basic operation of the Bind Control Model using these four types of nodes. For example, the motion sensor event bounded to the alarm clock bell realizes instant security system, which tells us intrusion of someone by ringing the bell.

3 Battery-less Technology

3.1 Background

In many researches on sensor network systems, a sensor network node is usually assumed as a battery-powerd device. And in many cases, such device is considered as a disposable device, and is assumed to be scattered in various environments. Based on these assumptions, a lot of battery-aware communication protocols have been proposed. However we believe that such battery-powered and disposable devices are not acceptable for practical sensor network applications.

In many consumer applications, there are few applications utilizing disposable sensor devices. Usually, disposable devices are only acceptable for military or disaster applications. For example, if we apply sensor network for monitoring temperature in a large plastic green house, we may not leave lots of dead sensor network devices in the environment because such devices and dead battery become obstacles for farm work and are harmful for products. This means that we have to replace or recharge huge amount of dead batteries if we apply battery-powered wireless sensor network systems for various consumer applications. Needless to say, doing such maintenance is unrealistic.

On the other hand, there have been developed lots of technologies for obtaining electric power without batteries. For example, we can obtain electric power from light, wind, heat, and vibration through various conversion devices. If we can utilize such energy sources for driving sensor network devices, we need not to care about battery problem in sensor network systems.

From this point of view, we have developed a prototype of battery-less sensor network system called “ Solar Biscuit ” . The Solar Biscuit system consists of many Solar Biscuit devices, and each device employs a super capacitor equipped with a small solar cell for its energy source.

The major difference between the Solar Biscuit device and conventional sensor network device is that the device has rechargeable energy source. Hence, communication mechanism of the Solar Biscuit system must be designed by considering not only power consumption but also power charging.

3.2 Solar Biscuit Design

The Solar Biscuit device implements a one-chip MPU (MICROCHIP PIC 18LF452, 7.3728 MHz), a 315 MHz RF module (CHIPCON CC1000), and a Temperature and Humidity sensor (SENSIRION SH11) on a 5 cm x 5 cm main circuit board. A 5 V 1.0 F super capacitor (NEC TOKIN, FT0H105Z) equipped with A 5 cm x 5 cm solar cell (SHELL SOLAR, Monolithic Silicon Solar Cell) provides energy to the device. The MPU in the Solar Biscuit device can measure the charge level of super capacitor through an internal A/D conversion circuit. If the charge level is less than 3.5 V, a CMOS reset IC (ROHM BD4835G) shuts down the microcontroller (i.e., the MPU must enter sleep mode before charge level becomes less than 3.5 V). If the charge level becomes 3.675 V, the reset IC restarts the MPU.

Our interest in designing the Solar Biscuit system is to develop a novel communication mechanism for battery-less sensor network system based on requirements from practical consumer applications. At the start of designing such communication mechanism, we set the target application of the Solar Biscuit system. In our opinion, the most suitable application of the Solar Biscuit system is environmental moni-

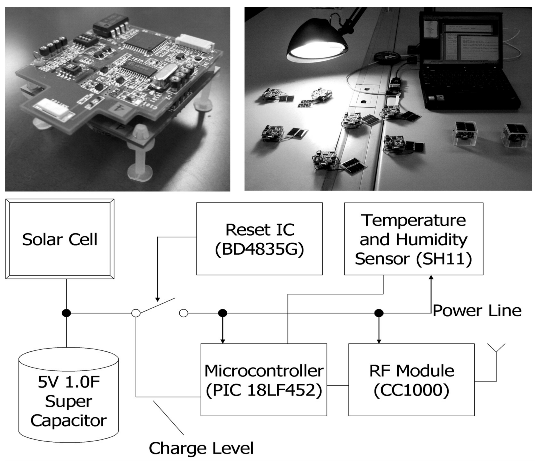

Figure 4: Hardware Implementation

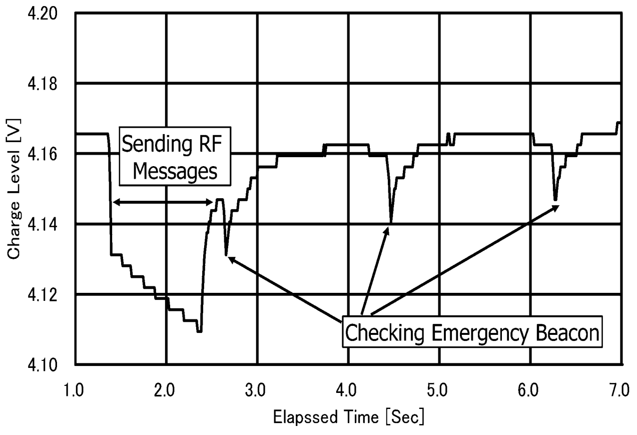

Figure 6: Example Communication Cycle

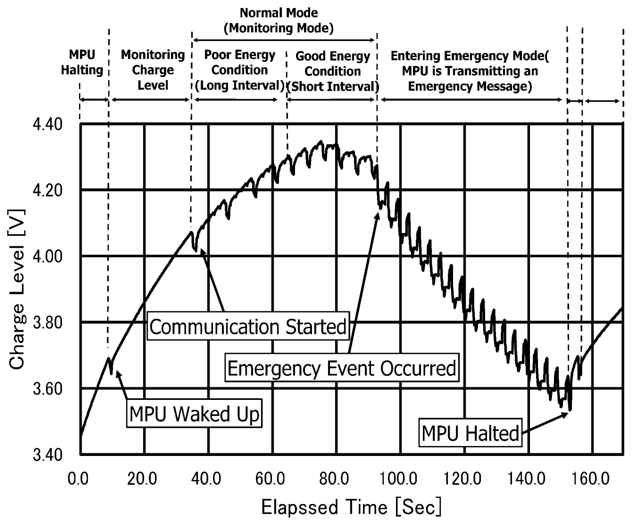

Figure 5: Behavior of Solar Biscuit Node

toring application. Usually such application does not require high data rate and is not so sensitive to delay in normal operation. For example, let's consider the system reports temperature in a plastic green house in a certain interval. In this case lack of several data packets is not so important problem. However, in an emergency situation (e.g. unusual temperature is detected at a device), the system must send this important event to a sink as quick as possible.

3.3 Communication Strategy

In order to support such kind of application, we designed an elementary communication strategy, which has both normal and emergency modes operation, for Solar Biscuit devices. In this communication strategy, we assume that a Solar Biscuit device can decide the optimal time ratio of both active and sleep mode based on provided power from solar cell (Figure 5). Here, the active mode means MPU and RF module are in active mode. Sleep mode means MPU and RF module are in low-power sleep mode, but awakes for a short period of time to hear emergency message (Figure 6). Note that each Solar Biscuit device charges its super capacitor in sleep mode.

The key point in the normal mode operation is to maximize network availability by randomizing sleep time at each device. As shown in Figure 7, when a device awakes from sleep mode, it sends a " good morning message " to neighboring devices. If an active device hears messages from more than predefined number of active device, it extends its interval of sleep time based on a certain probability. When a device has a data to send to a sink node, it simply floods the data to active devices. Thus, the data is forwarded to the sink by active devices in turn. The data has a Time-To-Live (TTL), and if the TTL expires, the data will be dropped.

On the other hand, once an emergency event is occurred, the system enters emergency mode operation. In this case, a node detecting the emergency event awakes all nodes in the system by sending an emergency message. After that, an emergency data is sent to a sink via flooding.

Untill now, we have just finished implementation of hardware and communication software of the Solar Biscuit system. Although our implementation is basically successful,more detailed evaluation is required to make system more practical. Currently, we are measureing various characteristics of the system including error ratio, throughput, and availability of the Solar Biscuit network to obtain good feedbacks for designing better communication protocol.

4 Precise Localization

4.1 Background

For sensor network systems, physical location of sensor network node is one of the key information to support various applications. This is because sensor network systems deal information which is highly depending on physical space. In other words, without location information the information

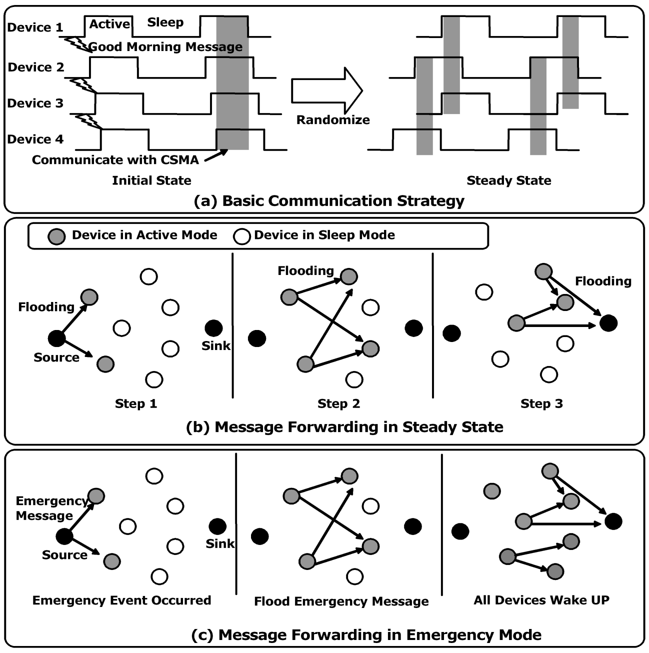

Figure 7: Communication Strategy

obtained from sensor network is meaningless.

Usually, we can easily obtain precise location information in outdoor environments by using the global positioning system (GPS). However, applications, such as factory automation, plant monitoring, and indoor context-aware application, require more precise location information. Moreover, the GPS is usually available only in outdoor environment. Therefore, researches on location systems have focused mainly on systems that can provide more precise location information and that is available in indoor environment.

To archive such goals, several positioning systems have been proposed. Active Bat[7] and Cricket[8] use ultrasonic pulse TDOA (Time Difference of Arrival) to measure high precision 3D position and orientation in indoor environment, but they require an extensive hardware infrastructure. However, such systems usually require manual pre-configurations of the locations of reference beacons or sensors. The setup and management costs would be unacceptably high if we apply them to large scale environment such as an office building. Ad-hoc localization mechanism described in [9] can be applied to such problem. In [9], the authors proposed collaborative multilateration algorithm to solve localization problem in a distributed manner, and performed detailed simulation-based analysis of a distributed localization system. To design practical location information infrastructure, we believe that experimental analysis is also needed to discover practical problem in distributed localization system.

From this point of view, we have developed a distributed positioning system called DOLPHIN (Distributed Object Localization System for Physical-space Internetworking) that can determine objects ' position using only few manually

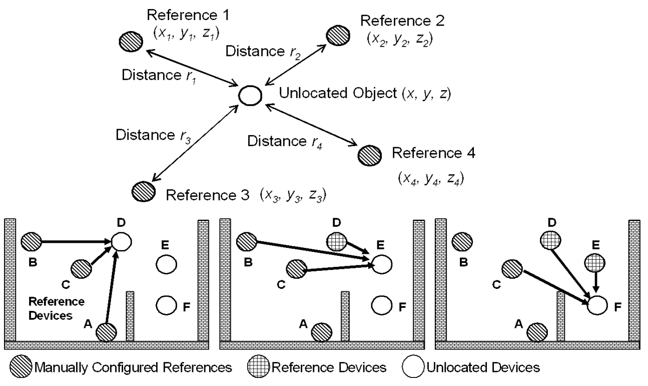
Reference 1 (x_1, y_1, z_1)
Reference 2 (x_2, y_2, z_2)
Distance r_2
Distance r_1
Unlocated Object (x, y, z)
Distance r_3
Distance r_4
Reference 4 (x_4, y_4, z_4)
Reference 3 (x_3, y_3, z_3)
Reference Devices
A
B
C
D
E
F
Manually Configured References
Reference Devices
Unlocated Devices

Figure 8: Iterative Multilateration

configured references. The system is made from off-the-shelf hardware devices, and implements a simple but practical distributed positioning algorithm.

4.2 DOLPHN Design

Basically, the DOLPHIN system determines location of various object based on a TOA (Time of Arrival) positioning technique. Figure 1 illustrates the principle of the TOA positioning. In order to determine an unlocated object position in three dimensions (x, y, z), ultrasonic-based distance measurements are made to four references resulting in the following equations:

$$r_i = \sqrt{(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2} \qquad (1)$$

where (x_i, y_i, z_i) denotes the position of the i-th reference, and r_i is the distance between unlocated object and the reference. The distance is measured by using TOA of ultrasonic pulses, and RF signal is usually used for time synchronization. The position of the unlocated device is determined by solving the above equation for four or more references.

Based on this positioning principle, we apply the idea of iterative multilateration to our system. Figure 8 illustrates the basic idea of the iterative multilateration technique. In the initial state, devices A, B and C have precisely measured positions, and the positions of the other devices are unknown. In the next step, device D, which can directly receive signals from devices A, B and C, can determine its position (here, we assume that one device can compute its position by receiving three or more signals from the references). However, devices E and F cannot yet receive a sufficient number of signals to determine their position due to such physical obstacles as the wall. Here, if the position of device D is determined and device E can receive a signal from device D, device E can compute its position by using signals from devices B, C and D. If the locations of device D and E are determined, device F can compute its position using devices C, D and E. In this way, all devices can be located.

However, iterative multilateration is just an idea to locate huge numbers of objects by using a small number of references. To develop a practical system, we should consider not

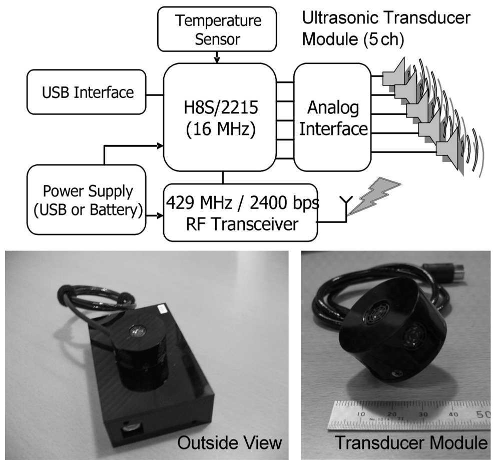
Temperature Sensor
Ultrasonic Transducer Module (5ch)
USB Interface
H8S/2215 (16 MHz)
Analog Interface
Power Supply (USB or Battery)
429 MHz / 2400 bps RF Transceiver
Outside View
Transducer Module

Figure 9: Implementation of DOLPHIN systemy

only how to apply iterative multilateration to our system, but also how to reduce positioning error that occurs in actual environment. To find error factors in iterative multilateration, we have implemented and tested the DOLPHIN system.

4.3 Implementation and Evaluation

Figure 9 illustrates a block diagram of the DOLPHIN device. To implement iterative multilateration, bi-directional ultrasonic transducers are used for both sending and receiving ultrasonic signals. As shown in the bottom of Figure 9, we attached five transducers to the cylindrical module to extend the coverage of the ultrasonic signals in every direction. The DOLPHIN device also has a 2400-bps, 10-channel, 429-MHz RF transceiver for time synchronization and the exchange of control messages. A one-chip microcontroller controls both the ultrasonic transducer and the RF transceiver, and calculates a device ’s position by performing multilateration. The microcontroller detects the received signal strength information of the ultrasonic signal at each transducer via internal A/D converters. This implementation enables us to estimate angle of arrival of ultrasonic signal based on received signal strength at each transducer. This CPU has a USB interface, and we can pull location data out through the USB interface as well as supply power to the device by attaching the device to a USB-enabled device such as a laptop or a PDA.

After implementing the system, we have investigated error factors of our system in various situations of an actual room environment. Through our experimentations, we found that there were two major factors leading to errors in the DOLPHIN system: error accumulation and no-line-of-sight propagation of ultrasonic signals. The error accumulation is caused by the structural characteristic of iterative multilateration, in which an unlocated node that can estimate its position using references becomes a new reference. Hence, the new reference accumulate positioning errors of references that are used by the new reference. The no-line-of-sight propagation problem occurs when there are many obstacles in the

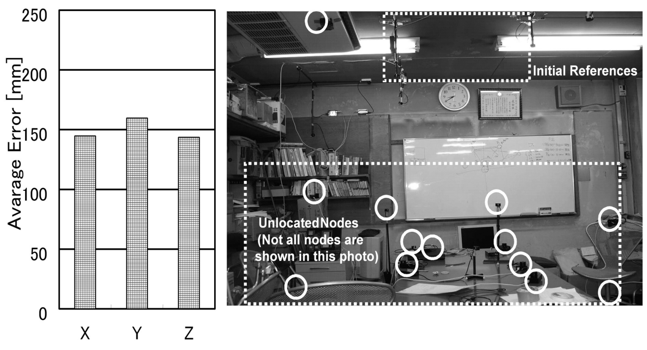

Figure 10: Positioning Accuracy

room. The no-line-of-sight signal affects desitance measurement in the DOLPHIN system and causes serious positioning error. To mitigate effects of these error sources, we have developped priority-based reference selection algorithm and geometry-based no-line-of-sight signal rejection technique. Although we do not describe the details of such techniques in this paper (see [10]), it is important that practical solution for this kind of problems can be found in implementation-based approach.

Figure 10 shows experimental result of the DOLPHIN system. We evaluated the positioning accuracy in a room environment. we distributed 24 devices in a small room in our laboratory, and utilized 4 references and 20 unlocated devices to investigate how the number of references affects positioning accuracy. In this experiment, we applied all techniques for achieving high accuracy positioning (i.e. techniques for mitigation of error accumulation and no-line-of-sight mulitipath error). The results showed that the DOLPHIN system could determine objects ’ position with an accuracy of less than 20 cm in actual indoor environment. If we do not apply techniques for achieving high accuracy positioning, error accumulation and no-line-of-sight propagation seriously degrades positioning accuracy, and it results in the accuracy of over 2 m.

5 Summary

This paper introduced how we have tackled to design practical sensor network system. We stated that development of all flexible testbed, battery-less technology, and precise localization technique is quite important for practical sensor network systems. As our answers to this challenge, we introduced three systems, PAVENET, Solar Biscuit, and DOLPHIN, including design philosophy of each system.

Though implementation-based approach requires much more effort and time than simulation-based approach, we believe that it is through implementation that practical sensor network can be truly designed.

References

[1] I. F. Akyildiz, et al.„ “A Survey on Sensor Networks”, IEEE Communications Magazine, 40(8), pp102-114, 2002.

[2] Crossbow Wireless Sensor Networks, http://www.xbow.com/.

[3] M. Beigl, et al., “Smart-Its: An Embedded Platform for Smart Objects”, Smart Objects Conference (sOc), May 2003.

[4] D. Estrin, et al., “Next Century Challenges: Scalable Coordination in Sensor Networks,” Proc. of MobiCOM1999, Aug. 1999.

[5] S. Saruwatari, et al., “Pavenet: A Hardware and Software Framework For Wireless Sensor Networks,” Proc. of INSS2004, June 2004.

[6] T. Kashima, et al., “A Bind Control Model For Real-space Programming in Ubiquitous Computing Environment,” UBICOMP2004 poster, Sept. 2004.

[7] A.Ward, et al., “ A New Location Technique for the Active Office”, IEEE Personal Communications Magazine, Vol. 4, No. 5, October 1997.

[8] N.Priyantha, et al. “The Cricket Compass for Context-aware Mobile Applications”, Proc. MOBICOM2001, July 2001.

[9] A. Savvides, et al., “Dynamic Fine Grained Localization in Ad-Hoc Sensor Networks”, Proc. MOBICOM2001, July 2001.

[10] M. Minami et al., “DOLPHIN: A Practical Approach for Implementing a Fully Distributed Indoor Ultrasonic Positioning System”, Proc. UBICOMP2004, Sept. 2004.