Design and Implementation of a Sensor Network Node for Ubiquitous Computing Environment

Yoshihiro KAWAHARA, Masateru MINAMI, Hiroyuki MORIKAWA, Tomonori AOYAMA
Bldg. #3, EE Dept., The University of Tokyo, 7-3-1 Hongo Bunkyo-ku, Tokyo, JAPAN
Phone: +81-3-5841-6710 / FAX: +81-3-5841-6776
{kawahara, minami, mori, aoyama}@mlab.t.u-tokyo.ac.jp

***Abstract*— "Smart" and "attentive" services using users' context are going to be realized in future ubiquitous computing environment. Sensor network is receiving considerable attention as one of the key technologies to keep track of the users' context in the real world. Conventional sensor network research mainly aimed at environmental monitoring. However requirements of the sensor network infrastructure for ubiquitous computing is different from the one for conventional environmental monitoring. We have developed a sensor network testbed named "U^3" to investigate issues that could arise when developing novel applications in a ubiquitous computing environment. U^3 is a wireless sensor node that is capable of communicating with other nodes to carry sensing data and queries issued by users. In this paper we describe hardware and software architecture of U^3 and present the performance evaluation of a sample application developed on U^3.**

Keywords-component; Sensor Network, Ubiquitous Computing, U^3

I. Introduction

Ubiquitous computing is a technology aimed at supporting human activities with a number of computers and sensors that are deeply integrated into our daily lives. Such "smart" and "attentive" services would be realized using users' context and preferences in the real world. Sensor network is a key technology to obtain users' contexts in support of such devices.

A large number of researchers have been working on sensor network technologies primarily for the realization of large-scale environmental monitoring systems for military use and/or scientific use [1-6]. Requirements of the sensor network infrastructure for ubiquitous computing are different from the one for conventional environmental monitoring [7]. When designing practical sensor network architecture for future ubiquitous computing environment, it is desirable that requirements of the applications be made clear. However, it is so far difficult to envision how such future applications should be like. We would like to have an effective way to implement and evaluate prototype applications as well as to clarify technical challenges. To that end, we have developed a sensor network development testbed "U^3 (U-Cube)" that allows developers to flexibly implement various applications. U^3 is a 50mm cube that contains a power module, a CPU module, an RF communication module, and a sensor module. The first implementation of U^3 is capable of sensing several types of data such as temperature, brightness, and the presence of human and sending this information to other nodes and/or peripheral devices including PC and PDA.

In this paper, we present hardware and software architectures of U^3 in the next section. Then we describe the performance evaluation of a sample application developed on U^3. Before concluding this paper, we describe some related works and compatibility issues.

II. Design of Hardware and Software

A. Hardware

There is a wide range of potential applications and protocols for sensor networks. For environmental monitoring applications, it would be useful if the wireless nodes provide generic sensor interface so that the sensors can be easily replaced. Due to practical reasons, it would be nice to be able to provide solar panel and rechargeable batteries. Moreover, users might want to choose appropriate CPU and wireless communication devices according to the power consumption and required processing/transmission speed. To meet these kinds of requirements, hardware components of sensor node should be constructed as an ensemble of independent modules that can be replaced easily. However, conventional sensor network nodes [8-10], such as MICA Mote [11,12], only allow replacement of sensor board. Other components such as CPU and wireless communication module, are not replaceable.

To this end, we divided the functionality of the sensor node into four physically separated modules: power control module, processing module, communication module and sensing/actuating module. Each module is connected to each other by a bus connector to achieve extensibility.

B. Software

Due to the sensor nodes' reduced physical size and restricted power consumption, software on the sensor nodes hardware must make efficient use of processor and memory while enabling low power communication. In this section, we describe task scheduling and API layering in the sensor node.

1) Event and task scheduling

In general, one sensor node plays several roles in the sensor networks. For example, information may be simultaneously captured from sensors, manipulated, and streamed onto a network. Alternatively, data may be received from other nodes and forwarded in multi-hop routing or bridging situations [13]. To realize such concurrent execution within resource-restricted hardware, we allow interruption of tasks by events. Here, an event is defined as a process which has to be executed immediately and is assumed to complete immediately. Event includes the arrival of wireless packets, etc. On the other hand, a task is defined as a process which takes longer than an event to finish. It includes periodic capturing of environmental data, etc. Event-based task scheduling needs low overhead for state transition compared to stack based approach. Accordingly this scheme saves CPU load and power consumption.

MAC (Media Access Control) algorithm significantly affects system performance of sensor node. Specifically, accurate synchronization between sending and receiving nodes is crucial for high speed and reliable transfer. Accordingly, application tasks can be frequently interrupted by wireless communication events. To remove such loads from the CPU, our sensor node provides separate dedicated CPU for wireless communication and application tasks.

2) Functional layering

The primary objective of U^3 is to provide a testbed sensor network for ubiquitous computing environment. Having a testbed development environment is essential for attractive application to be produced. For instance, when a user wants to design appropriate communication protocol for a specific application, it would be useful if various APIs for controlling communication functionality are provided in the development kit. Though TinyOS [14] is a component-based runtime environment designed to provide support for embedded systems such as the MICA motes, they do not provide separate communication module. So far, we have defined several APIs that provide layered abstracted communication devices (application, media access, physical layer of RF and IrDA 1.0). This layering enables developers not only to reuse the APIs but also to concentrate on the development and evaluation of specific function or protocol they are interested in. Of course, it is conceivable that conventional layering no longer holds for sensor networking. Even in that case, however, developers can choose to do without the APIs and conventional layering.

III. IMPLEMENTATION

A. Hardware

Figure 1 and Table 1 show U^3 hardware components and module names, respectively. The hardware components are commodity off-the-shelf products. In U^3, 4 independent functional boards are interconnected with 26-pin / 2.54mm (standard) pitch bus connector. I2C bus protocol has been implemented for communicating among components.

Power module: the power module contains power management device including three rechargeable AAA batteries. Recharging power source can be supplied by an AC adapter or a solar panel.

CPU module: the CPU module is used for controlling the power source (selection of the power source, recharging the batteries), storing and processing messages sent by other nodes, controlling the sensor and actuator devices, etc. PIC18F452 (Microchip) is adopted as the processor. IrDA 1.0 communication interface in the CPU module is used as an interface between U^3 node and peripheral devices such as a PC or a PDA. Application programs and sensor data can be uploaded and/or downloaded through this connection.

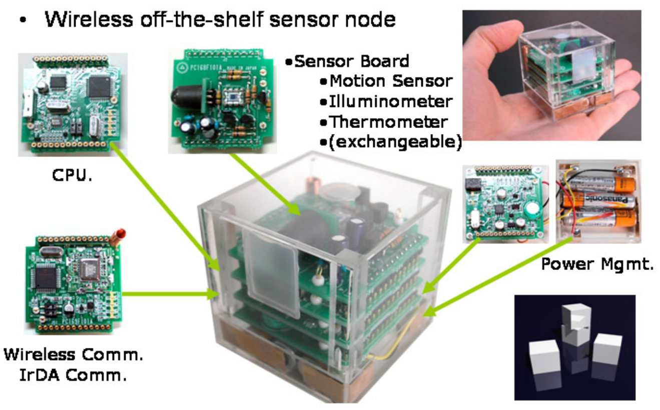

Figure 1. Hardware components

Table 1. Major components

Sensor		Motion, Photo, Temperature
RF Transceiver		CDC-TR-02B(315MHz)
Micro Controller Unit		PIC18F452
IrDA1.0	Controller	MCP2150
	Transceiver	RPM851A
Calender		RX-8564CF
Battery		Ni-MH

Communication Module: this module consists of an RF Monolithics transceiver (300MHz band, On-Off keying, 115.2kbps), helical antenna and second PIC microcontroller for processing network protocols. The current implementation realizes data transmission at up to 100kbps with our current implementation of CSMA/CA protocol. The transmission range is within a radius of 30m. On the communication module, a dedicated CPU is used to drive RF transceiver. As mentioned in the previous section, we can implement various MAC and network layer protocols on this CPU.

Sensor Module: the sensor module is a module for obtaining information of the real world. Though the current node implements only a motion sensor, a brightness sensor, and a thermometer, various transducers can be connected to the board via generic bus connector.

B. Software

Figure 2 shows the software architecture of U^3. We adopted the C language as the programming language because several commercial C compilers such as the CCS C Compiler provides plenty of built-in functions that handle interrupts, etc.

U^3 software consists of the application software, the wireless communication software, and the development environment. The application software and the wireless communication software are implemented in PICs on the CPU board and the wireless communication board respectively. Messages between the software traveled the I2C bus. Several major APIs are listed in Table 2.

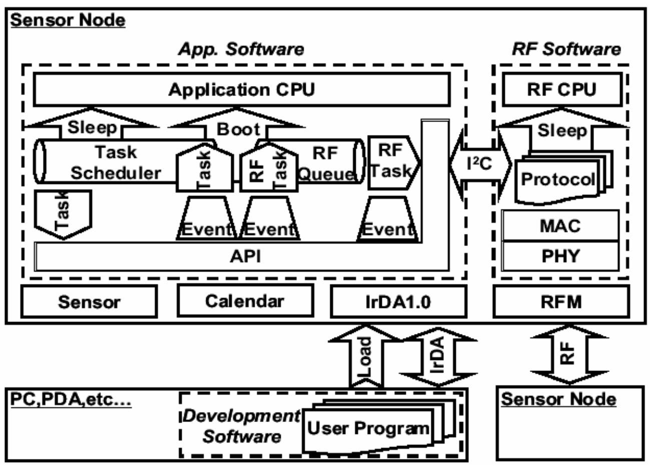

Figure 2. Software architecture

Table 2. API

Application API
`void post_task(int msg_id);`
`// Launch a task named` *`msg_id`*
`void irda_send(packet_type* packet);`
`// send packet with IrDA1.0`
`void packet_send(packet_type* packet);`
`// send packet to wireless comm. device`
`void on_timer(void);`
`// define periodical event`
`void packet_on_receive(packet_type* packet);`
`// invoked on receiving packet from`
`// wireless comm. device`
Communication API
`void to_app_layer(packet_type* packet);`
`// send data to application CPU via I2C`
`void from_app_layer(packet_type* packet);`
`// receive data from application CPU via I2C`
`void to_mac_layer(packet_type* packet);`
`// send packet to MAC layer interface`
`void from_mac_layer(packet_type* packet);`
`// receive packet from MAC layer interface`
`void to_physical_layer(packet_type* packet);`
`// send packet to RF module`
`int8 from_physical_layer`
`(packet_type* packet, int8 time);`
`// receive packet at` *`time`*
`void rf_sleep_mode(void);`
`// turn off RF module`
`int1 carrier_sense(void);`
`// turn on RF module to check radio signal`

1) Application software

Application softwares control the sensor and actuating devices, and forwards processed data to the wireless communication software. Multiple tasks are scheduled in the FIFO-based task scheduler. Application softwares also provide APIs for communicating with a PC or PDA (via IrDA), or other nodes (via the RF module). After all the tasks in the scheduler have been processed, the system turns the CPU into sleep mode to save power. Timer events generated by the calendar IC and packet arrival events generated by the IrDA chip and the wireless communication software can wake up the CPU.

2) Wireless communication software

The wireless communication software provides APIs that bridge application CPU and the wireless RF modules. Inside wireless communication software, routing protocol and MAC protocol are independently implemented so that each of the protocol can be replaced according to users' requirements. In the physical layer, we provide periodic transmission and reception control of the RF communication module. As the RF module transmits data by ASK (Amplitude Shift Keying), we encode each bit with the Manchester Code.[1] This encoded data is sent after preamble (1 Byte) and header (2 Bytes) are sent.

3) Development environment

For the development environment, we provide a program loader that helps users upload application programs into U^3 via IrDA 1.0. This program loader is also used as the interface between the sensor node and external devices such as a PC or a PDA. Through the connection, external device can send query to the sensor node and pull out captured data.

4) Packet format

```
typedef struct
{
uint16  addr;   // destination address
uint8   type;   // Application ID
uint8   group;  // Network ID
uint8   length; // Packet Length
uint8   data[DATA_LENGTH];
                //Payload (max:29bytes)
uint16 crc;     // CRC
} packet_type
```

Figure 3. Packet Format

In U^3, all the messages exchanged between nodes, and between a nodes and external devices use the same packet format (Figure 3). The packet contains destination address (1 byte), application type (1 byte), network ID (1 byte), packet length (1 byte), payload (Max: 29 bytes), and CRC (2 bytes). Query data, captured data, etc. are put in the payload.

IV. A SAMPLE APPLICATION

A. A sample application

We have evaluated the performance of the U^3 sensor node by developing a simple application that detects human motion in a room. As an experiment, we measured the success ratio of packet transmission. We deployed six U^3 nodes in a 9m x 12m indoor environment. Each node is given a numbered 2-byte ID. One of the nodes, called "sink node," is connected to a data collection PC via IrDA. The sink node periodically queries the other nodes (source nodes) about motion sensor data over the RF wireless link. In response to the query, source nodes report to the sink node whether they detected a human motion. As

[1] A Manchester code is a code in which data and clock signals are combined to form a single self-synchronizing data stream. Each encoded bit contains a transition at the midpoint of a bit period, the direction of transition determines whether the bit is a "0" or a "1."

queries are periodically generated by the sink node, each source node discards old queries when it receives a new one. When sink node receives a report from a source node, it transfers the data to the PC to visualize the sensed data. To achieve redundancy of multi-hop wireless network, queries and responses are disseminated by flooding. Collision avoidance and multiple accesses over RF wireless link is achieved by implementing the CSMA/CA protocol.

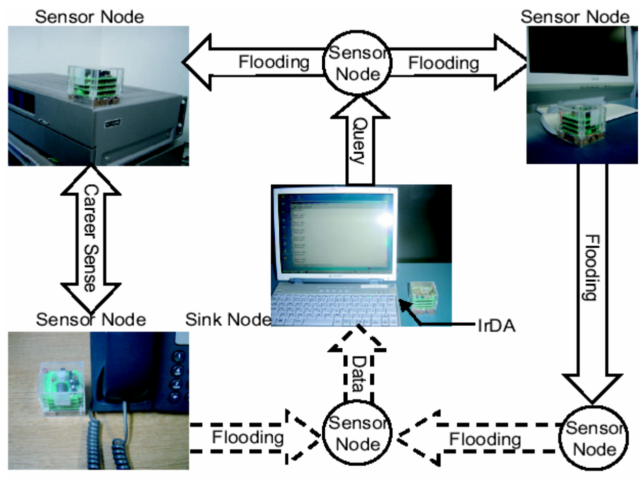

Figure 4. Experimental application

B. Performance evaluation

In this application, the sink node periodically queries all the nodes in the network. Therefore fairness of successful transmission rate and sink to source latency are important metrics. Figure 5 shows each node's rate of successful transmission rate when the query repetition period is 0.5 msec, 1.0 sec, and 1.5 sec. The result shows that successful transmission rate is sufficiently high (91%) even when the query repetition period is 1.5 sec. This is a convergence of discarding sensor data packets on intermediate nodes if the nodes receive newer queries.

Fairness of the successful transmission varies widely when the repetition period is large. This phenomenon results from the effect of packet loss over multi-hop networks.

Figure 6 shows the success rate of data transfer when the number of nodes changes. In this experiment, each node tries to transmit a response packet that contains sensor data at random intervals over 100 seconds. Each packet is 13 bytes in length (including header and payload). Figure 4 shows that the success rate of the transmission varies widely as the number of nodes increases. This is due to the unbalanced random value used for the CSMA/CA back-off operation. As the operation to generate good random numbers is too expensive for low power microprocessor, we use only a simply random function for the CSMA/CA backoff operation.

Considering this experiment as a simple check-out for the entire U3 sensor node, we have not made any effort to improve the network protocols (e.g. with regards to fairness, delay, etc.). Each protocol or module must be improved according to the requirements of each application. For this reason, it is essential to provide a sensor network testbed that allows for the replacement of hardware and software modules as needed.

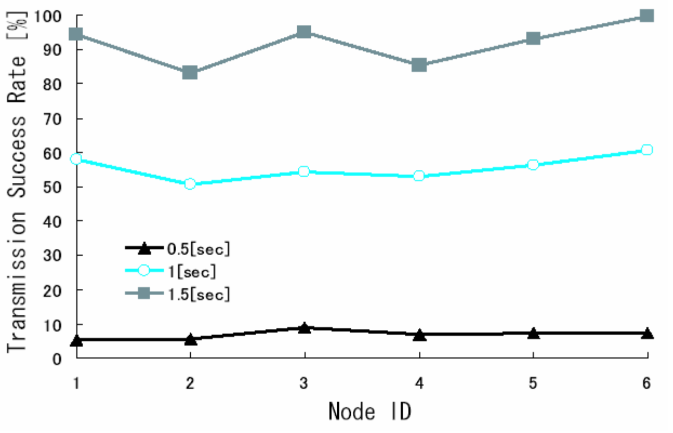

Figure 5. Transmission Success Rate

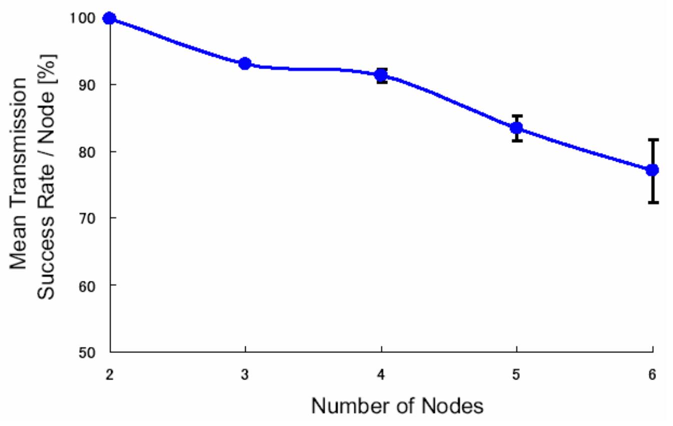

Figure 6. Mean Transmission Success Rate

V. RELATED WORKS AND COMPATIBILITY

MICA Motes (developed at UC Berkeley) and U^3 shares similar motivation: to develop off-the-shelf sensor network testbed. However, they are different in several points. Although MICA and U^3 use similar RF wireless communication module for communication between nodes, the frequencies used are different due to different legal regulation. The transmission range of frequency bandwidth 916MHz used by MICA is strictly restricted to about 1 meter in some countries including Japan. In U^3, we use two PIC microcontrollers for processor and network controller. Compared to the ATMEL 90LS8535 installed in MICA, PIC has advantage in power consumption especially in "sleep" mode. Due to the use of these two controllers, U^3 has a potential advantage in power consumption. In addition, MICA is equipped with replaceable sensor board, in U^3 not only the sensor board but also the CPU, network and power modules can be replaced. IrDA communication with other peripheral device is also unique to U^3. As for software architecture, TinyOS for MICA also provides event-based development environments. However, they do not provide layering of the network stacks: they provide a monolithic networking module.

In the near future, a variety of sensor nodes that consist of different hardware architectures would emerge. In such a situation, compatibility of the sensor nodes would be required. To this end, we use the same message format as MICA and developed a gateway that bridges the wireless communication both MICA mote and U^3.

VI. CONCLUSION AND FUTURE WORK

In this paper, we presented a design and implementation of a wireless sensor network node, U^3. U^3 is effective for development and evaluation of prototype applications because of the modularity and flexibility of both hardware and software. Now we are designing several extension modules including display module, other sensors, and so on.

REFERENCES

[1] J. Kahn, R. Katz, and K. Pister, "Next Century Challenges:Mobile Networking for "Smart Dust"," In Proceedings ofthe ACM/IEEE International Conference on Mobile Computingand Networking, pp. 271–278, Seattle, USA, Aug.1999.

[2] W. Heinzelman, J. Kulik, and H. Balakrishnan, "AdaptiveProtocols for Information Dissemination in Wireless SensorNetworks," In Proceedings of the ACM/IEEE InternationalConference on Mobile Computing and Networking, pp. 174–185 Seattle, USA, Aug. 1999.

[3] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan,"Energy-efficient communication protocol for wireless microsensornetworks," In Proceedings of the Hawaii InternationalConference on System Sciences, pp. 4–7 Maui, USA,Jan. 2000.

[4] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directeddiffusion: A scalable and robust communication paradigmfor sensor networks," In Proceedings of the ACM/IEEE InternationalConference on Mobile Computing and Networking,pp. 56–67, Boston, USA, Aug. 2000.

[5] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris,"Span: An Energy-Efficient Coordination Algorithm forTopology Maintenance in Ad Hoc Wireless Networks," InProceedings of the ACM/IEEE International Conference onMobile Computing and Networking, pp. 70–84, Rome, Italy,Jul. 2001.

[6] W. Ye, J. Heidemann and D. Estrin, "An Energy-EfficientMAC Protocol for Wireless Sensor Networks," In Proceedingsof the 21st International Annual Joint Conference ofthe IEEE Computer and Communications Societies, NewYork, USA, Jun. 2002.

[7] J. Hill, R. Szewczyk, A. Woo, S. Hollar, and D. Culler, andK. Pister, "System architecture directions for networkedsensors," In Proceedings of the 9th International Conferenceon Architectural Support for Programming Languages andOperating Systems, vol. 35, no. 11, pp. 93–104, Cambridge,USA, Nov. 2000.

[8] G. Pottie and W. Kaiser, "Wireless Integrated Network Sensors,"Communications of the ACM, vol. 43, pp. 51–58, May2000.

[9] SCADDS(Scalable Coordination Architectures for DeeplyDistributed Systems) http://www.isi.edu/scadds/

[10] WEBS(Wireless Embedded Systems) http://webs.cs.berkeley.edu/

[11] S. Hollar, "COTS Dust," Masters Thesis, University ofCalifornia, Berkeley, USA, 2000. http://wwwbsac.eecs.berkeley.edu/ shollar/shollar thesis.pdf

[12] MICA http://www.xbow.com/Products/Product pdf files/Wireless pdf/MICA.pdf

[13] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, J. Anderson,"Wireless Sensor Networks for Habitat Monitoring,"In Proceedings of First ACM Workshop on Wireless SensorNetworks and Applications, Atlanta, USA, Sep. 2002.

[14] TinyOS http://webs.cs.berkeley.edu/tos/