Jan. 2005, Vol.20, No.1, pp.105–117 *J. Comput. Sci. & Technol.*

BLOSSOMS: Building Lightweight Optimized Sensor Systems on a Massive Scale

Wen Gao[1], Lionel M. Ni[2], Zhi-Wei Xu[1], S. C. Cheung[2], Li Cui[1], and Qiong Luo[2]

[1]*Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080, P.R. China*

[2]*Department of Computer Science, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon Hong Kong Special Administrative Region, P.R. China*

E-mail: {wgao, zxu, lcui}@ict.ac.cn; {ni, scc, luo}@cs.ust.hk

Received November 3, 2004; revised December 13, 2004.

Abstract As a joint effort between the Chinese Academy of Sciences and the Hong Kong University of Science and Technology, the BLOSSOMS sensor network project aims to identify research issues at all levels from practical applications down to the design of sensor nodes. In this project, a heterogeneous sensor array including different types of application-dependent sensors as well as monitoring sensors and intruding sensors are being developed. Application-dependent power-aware communication protocols are also being studied for communications among sensor nodes. An ontology-based middleware is built to relieve the burden of application developers from collecting, classifying and processing messy sensing contexts. This project is also developing a set of tools allowing researchers to model, simulate/emulate, analyze, and monitor various functions of sensor networks.

Keywords sensor network, sensor node, communication protocol, middleware, network simulation, network monitor

1 Introduction

Recent advances in wireless communication and hardware have enabled the dense deployment of distributed sensor networks and opened a wide range of application domains, such as environmental monitoring for endangered species or ecosystems of changes in light, temperature, pressure, acoustics; security surveillance preventing attacks in the forms of chemical, biological, or radiological weapons; target tracking of moving objects; and battlefield awareness[1]. All these applications may have information collected from sensor nodes to be sent to the gateway (sink) from time to time. By exploiting the sensor network's spatial coverage and multiplicity of sensing modalities, the network can achieve a good global measurement. Research in sensor networks has received a great deal of attention worldwide, where the most notable one is Berkeley's MOTE[2,3], due to its potential impact on so many application domains.

In March 2004, recognizing the importance of sensor networks, Chinese Academy of Sciences and the Hong Kong University of Science and Technology have jointly launched an effort to investigate both fundamental and practical research issues in sensor networks. The goal of this research is to build lightweight optimized sensor systems on a massive scale, namely the BLOSSOMS project. The objective of this research project is to identify research issues at all levels from practical applications down to the design of sensor nodes. As a multidiscipline research, this project involves researchers from different disciplines including hardware design, embedded systems, wired and wireless communications, software engineering, distributed query processing, machine learning, and performance evaluation. In a short period of time, the project has made a very good progress[4]. This paper will give an overview of the BLOSSOMS project and introduce some research directions being investigated.

2 Project Overview

Fig.1 shows the system architecture of the BLOSSOMS project. There are four layers. The bottom two layers are implemented in individual sensor nodes, while the top two layers are executed at gateway nodes or some application nodes. Typically, a gateway node will broadcast a command to all active sensor nodes and selected active sensor nodes will report the result back to the gateway node. The communication among the sensor nodes and gateway nodes is wireless and usually short ranged to reduce power consumption, such as Zig-

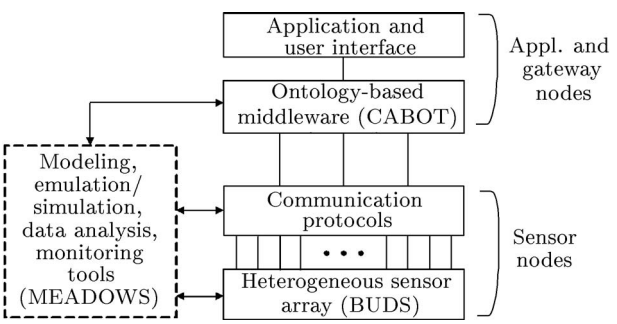


Fig.1. BLOSSOMS system architecture.

*Regular Paper

This work is supported in part by Grants 20046040 and 20034020 from ICT, CAS and Grants HKUST6161/03E and HKUST6158/03E from the Hong Kong Research Grants Council (RGC).

Bee. The communication among gateway nodes and application nodes can be either wired or wireless and typically adopts the standard protocols, such as 802.11x and TCP/UDP/IP. In addition, the MEADOWS module (on the left side of the figure) provides a set of tools for sensor network studies. The following subsections describe each BLOSSOMS component in more detail.

2.1 BUDS: Heterogeneous Sensor Nodes

A wireless sensor network consists of many sensor nodes deployed throughout an area of interest which link our physical world with the computing world. Sensor nodes are of various application specific functionalities. Many researchers are currently engaged in developing pervasive sensor nodes[2,5,6] due to the great promise and potential with applications shown by various wireless remote sensor network practice[1,7−11].

It is well understood that a sensor network node is an embedded system consisting of four basic components, namely a sensing unit, a processing unit, a communication unit, and a power unit. Additionally, there may be application-dependent, optional units such as a mobilizer, a location finder, and a power generator included[1].

The functions that a sensor node performs include:

• Sensing: sensors are coupled to a physical world, measuring the physical parameters;

• Processing and storing: the CPU processes the raw data perceived, handles data packaging and unpackaging; whilst the memory unit performs data storing;

• Communicating: using a wireless transmission method the transceiver deals with data stream dissemination and collection;

• Controlling: hardware control, software control and a wide range of application specific procedures and tasks' control.

The functionalities of each basic unit of a sensor node are summarized in details as follows:

Sensing units are usually made up of application specific sensors and ADCs (analog to digital converters), which digitalize the analog signals produced by the sensors when they sensed particular phenomenon. In some cases, an actuator is also needed. Obviously the front end smart sensors play a key role in sensor networks which provide fundamental raw data for the rest of the network to calculate and compute. Although MEMS technology has been making steady progress in the past decades, there is still large space for the further development of smart front end sensors. Among them, various chemical and biochemical sensors remain one of the most challenging sensor groups to be explored and developed, e.g., sensors to detect toxic or explosive trace in public areas, sensors for diagnostic analysis and sensors used under extreme conditions. New sensing principle, new sensing material and new sensor design need to be invented and adopted.

The processing unit is usually associated with an embedded operating system, a microcontroller and a storage part. It manages data acquisition, analyzes the raw sensing data and formulates answers to specific user requests. It also controls the communication and performs a wide variety of application specified tasks. Energy and cost are two key constraints for processing components. Nodes may have different types of processors for certain specific tasks. For example, some sensor node may need a more powerful processor to run than other common nodes. A small embedded operation system is another key issue for an embedded system. Besides the basic ability for process management and resource management, it may also possess the capability for software tailor and real time management, the ability to provide support for embedded middleware, network protocols and embedded database.

The transceiver connects the sensor node to the network. Usually each of the sensor nodes has the capability to transmit data to and receive data from another node and the sink. The latter may further communicate with the task manager via Internet (or Satellite) and information reaches the end user. It is well understood that a transceiver is the most power-consuming component of the node. Thus the study of multi-hop communications and power saving management is crucial in this content.

The power unit delivers power to all the working parts of the node. Because of the limited capacity of the power unit, e.g., the limited lifetime of a battery, the development of the power unit itself and the design of a power saving working mode of the sensor network remain some of the most important technical issues. For some applications, a solar battery may be used.

Additionally, a sensor node may have application dependent functional subunits such as a location finder, a mobilizer, a power generator and other special-purpose sensors. The nature or number of such subunits may vary, depending on the application needs. It is a very interesting area to be continuously exploited.

Our research on sensor node architecture is currently undertaking at ICT, CAS. The goal is to build multifunctional sensing nodes, called BUDS, targeting on applications such as environmental monitoring, intelligent transportation systems, precision agriculture, remote medical care, and public safety monitoring and notification system.

Efforts are made to tackle the following technical issues.

• The development of new types of sensors, e.g., chemical sensors for detecting trace explosives by employing new sensing principle and materials such as various function polymers and nano materials. This project is carried out in collaboration with Institute of Chemistry of CAS. Experimental results will be published sep-

arately.

• The design of special-purpose sensor nodes, such as image sensor nodes, monitoring sensor nodes, and intruding sensor nodes. Here the monitoring sensor node is proposed to be a special type to help debug and monitor the behavior of other sensor nodes. The intruding sensor node is used to simulate malicious sensors trying to attack a working sensor array.

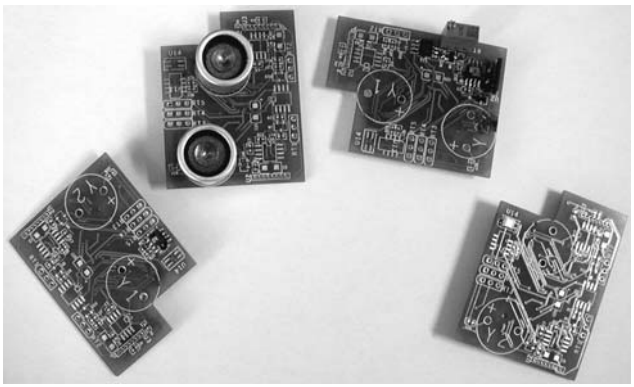


Fig.2. Photograph of a set of sensor board made of commercial sensors for temperature and humidity, distance, magnetic field, and light intensity detection (from left to right).

• The application of off-the-shelf sensors for a specific network construction, e.g., magnetic sensor, light sensor, temperature and humidity sensor, ultrasound sensor, pressure sensor as well as optical sensors. Fig.2 shows a photograph of a set of sensor board made of commercial sensors for light intensity, temperature, humidity, distance and magnetic field detection.

• Hardware and software co-design for low power management including embedded OS, application programs, microprocessor, RF module. System integration techniques, e.g., antenna design, product packaging and interfaces design. The photograph of Fig.3(a) shows a processing and communication board, where we use the open source TinyOS as the operation system. Fig.3(b) is a photograph of a complete sensor node. These models utilize Atmega128L micro-controller with 10-bit ADC. The RF communication chips we used are CC1000 from Chipcon, NRF905 and NRF401 from Nordic. In other biochemical sensor node designs for remote health care network, MSP430 from TI was used as micro-controller and CC2420 from Chipcon chosen as the communication chip which supports ZigBee protocol. By optimizing the design of antenna, the RF communication range of current node can reach 100 meters indoor and 300 meters outdoor. It can keep communication well across at least three floors inside the building. The frequency can be adjusted and fixed at 915/433/315 MHz, respectively.

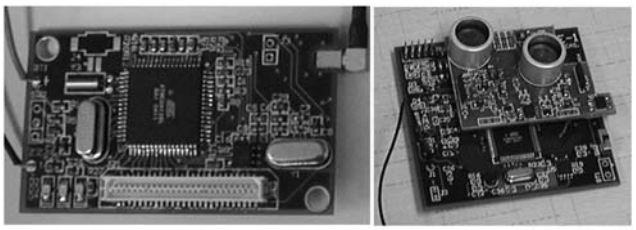


Fig.3. (a) Processing and communication boards. (b) Complete sensor node.

• The development of an embedded operating system for an ultra low power sensor node. Besides the basic ability for process control and resource management, it will also possess the capability for real time management, the ability to provide support for embedded middleware, network protocols and embedded database;

• Hardware and software co-design for system security management;

• Hardware and software co-design for sensor node mobilization and remote management. Fig.4(a) shows a photograph of a mobile sensor node. We have built a sensor node mobile carrier equipped with light sensors and ultrasound sensors and the driver control board is shown in Fig.4(b). There are two ways to control the move of the mobile carrier. Firstly, guided by the light and ultrasound sensors, the mobile carrier can move along a fixed path automatically, e.g., along a dark line on the floor as shown in Fig.4(a). Secondly, the moving direction and route of the mobile carrier can be remotely controlled by a program on end user's computer as illustrated in Fig.4(c) where the inserted picture shows the remote control program interface on a computer.

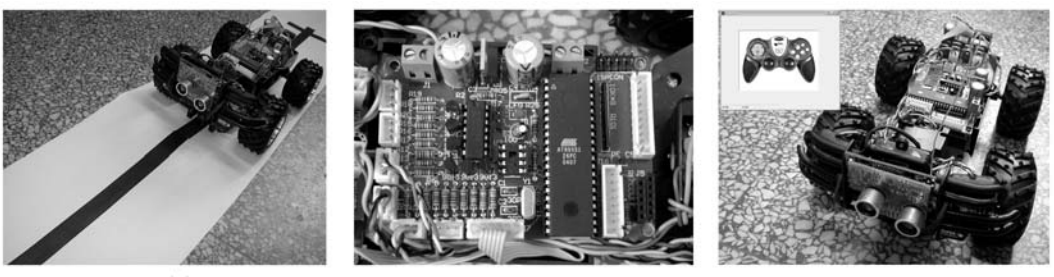


Fig.4. (a) Mobile sensor node carrier moving automatically along a fixed path. (b) Driver control board. (c) Mobile node carrier to be remotely controlled and the application program interface on end user's computer (inserted).

2.2 Communication Protocols

Routing protocols play a major role in enabling commands to be propagated from gateway nodes to sensor nodes and collecting sensed data of interest from sensor nodes back to the gateway nodes for further study. As is well known, sensors are very resource-constrained devices, which are powered by battery, have limited memory and possess low computational capability. Each sensor is prone to failure and the wireless communication is fairly unreliable. Thus, routing protocols for wireless sensor networks are required to work robustly and be able to deal with unpredictable changes in the network. The most important performance goal of routing protocols is to minimize the power consumption and hence postpone the lifetime of sensor networks. Other concerns of performance include connectivity of network, reliable data delivery and communication delay.

2.2.1 Routing Protocols for Sensor Networks

Routing protocols are very application dependent because different applications may have quite different communication patterns. Therefore, for a specific application, it is very important to identify the communication patterns involved so as to design an effective and customized routing protocol. In Fig.5, we summarize the possible communication patterns. There are two directions of data streams in sensor networks: sensor-to-gateway and gateway-to-sensor. In the sensor-to-gateway stream, data are originated in sensors and are going to be routed back to the gateway. The majority of data movements in sensor networks are of this type. In general, there are four different patterns, one-to-gateway, subset-to-gateway, region-to-gateway and all-to-gateway. A one-to-gateway communication happens when one sensor node has some information to report back to the gateway. A subset-to-gateway communication is needed when a subset of sensor nodes want to report information back to the gateway. The region-to-gateway communication is a special case of the subset-to-gateway communication, where those sensor nodes are physically near with each other and reside within the same designated region. When all active sensor nodes want to send data back, an all-to-gateway communication is required. In those communication patterns involving multiple sensors, they must be closely synchronized to allow data aggregation to support efficient communication. The converse direction of data stream is gateway-to-sensor, where the data to be sent is originated in the gateway. This kind of communication is usually needed when the gateway propagates commands (or queries) to sensors, or updates the software inside sensors. Likewise, there are four different patterns: gateway-to-one, gateway-to-subset, gateway-to-region and gateway-to-all.

We also observed the dual-operation phenomenon, namely, a gateway-to-sensor communication is usually immediately followed by a sensor-to-gateway communication, and vice versa. For example, the gateway issues a query which says "please report if the sensed temperature is greater than 70°C", which is a gateway-to-sensor communication. And then those sensors which sensed temperature higher than 70°C will route their data back to the gateway, which is a sensor-to-gateway communication. To the best of our knowledge, little research has been aware of dual-operation. We believe that more efficient routing protocols should take the dual-operation into account.

With the availability of the clear classification of communication patterns, we can get more insight into the dependency of routing protocols on specific applications. From the perspective of queries, for example, different queries require different communication patterns. Query "temperatures in Room 505" will invoke a region-to-gateway communication, and Query "temperatures above 80°C" will require a subset-to-gateway communication. Given the required communication patterns, routing protocols should be carefully designed accordingly. The ideal solution is that routing protocols are adaptive to communication patterns so that the protocols are most efficient and effective for the specific application.

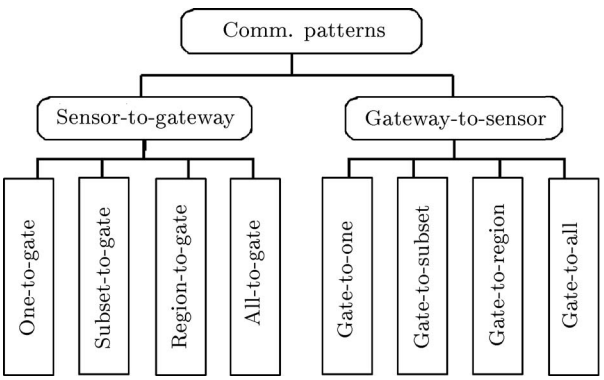


Fig.5. Communication patterns in sensor networks.

Current research for routing protocols[12−19] has been focused on sensor networks with one unique gateway only. It is very possible, however, that multiple gateways may be deployed in a large-scale sensor network for many purposes, such as better reliability. These gateways usually are connected by external high-speed connections. It makes big differences whether the gateways are elaborately coordinated. When there is no cooperation between the gateways, the whole of sensor nodes may be simply divided into several subsets and each gateway is in charge of a subset of sensors. If the gateways cooperate with each other, more sophisticated and effective communication can be achieved. In such a kind of sensor networks with multiple gateways, routing protocols will be very different from those for networks with only one gateway. Moreover, sensors nodes may be

heterogeneous in terms of sensing functions. The heterogeneity of sensors poses further challenges to the design of routing protocols.

The design of routing protocols also highly relies on availability of other information, such as physical location, global ID, etc. A good number of location-aided routing protocols[14,17−19] have been proposed, which hold the assumption that each sensor node has the accurate location information. GPS is a simple and direct solution to localization, but it is too costly for sensor networks due to the additional power consumption and high deployment expense. Thus, effective and inexpensive localization techniques have become very important, which is another topic of interest of our research. Global ID is desirable in senor networks so that each sensor can be distinguished from each other. For a resource-constrained sensor, however, it is not feasible to allocate a big address space for global ID, which will cause waste of memory and increased communication overhead. For operations of some routing protocols, we do need to distinguish sensor nodes to some extent, but a locally unique ID may be enough. Thus, this poses a challenging research opportunity.

In conclusion, in this project we aim to design a series of routing protocols for sensor networks, with the in-depth understanding that routing protocols are really application dependent, in particular, communication pattern dependent.

2.2.2 *Localization in Wireless Sensor Networks*

To achieve the appealing potential, location information of sensor nodes is very crucial for many applications of sensor networks. The term localization refers to the process of determining the physical location of every sensor node in a sensor network. If sensor nodes fail to obtain their locations, many applications would become infeasible. For instance, for event reporting applications, whenever an event is captured, the corresponding sensor node has to enclose its location in the event to be routed back to the sink; otherwise, the operator has no way to identify where the event occurred. And, as mentioned above, those location-aided routing protocols would become infeasible if there were no available location information in sensor nodes. Therefore, localization is really an important building block in sensor networks.

A good number of approaches[20−25] have been proposed for general location determination in pervasive computing. These approaches can be classified into different categories according to different criteria. (a) Infrastructure-based and ad-hoc. In an infrastructure-based scheme, clients cannot communicate with each other directly, or a client must first communicate with a base station. While in an ad-hoc scheme, a client can communicate with other nearby clients directly. (b) Map-based and non-map. A map-based approach builds a radio map in the offline phase and performs online location estimation based on the radio map. A non-map approach does not require any radio map built in the offline phase. (c) Anchor-based and anchor-free. Anchor-based approaches require that a number of anchors be pre-deployed across the sensor network, with the goal of computing absolute locations. Anchor-free approaches require no anchors and aim to obtain relative locations. (d) Mobile and static. Some approaches assume that clients are static, while other approaches tackle the mobility problem of clients. (e) Active and passive. In an active scheme, a client is able to compute its own location; while in a passive scheme, the location of a client cannot be computed by itself.

For wireless sensor networks, localization techniques based on radio frequency (RF) signals are very attractive due to their cost-effectiveness and flexibility. We can leverage the inherent radio-frequency (RF) communication capabilities of sensor nodes. Each sensor node is able to measure the strength of RF signals received from its neighboring sensors. One important characteristic of radio propagation is that the attenuation of the signal increases as the distance between the transmitter and receiver increases. Given the transmitting power, the loss of the signal strength can be calculated. Theoretical and empirical models can be used to translate the strength loss into a distance estimate.

It is highly challenging to accurately determine locations for the nodes in a sensor network. A sensor network is usually deployed across an unattended environment and is exposed to the unpredictable influences of the environment. In a large scale sensor network, a sensor node is typically a very resource-constrained device with small memory, limited computation capability, and more essentially, limited power. Any individual sensor node is prone to failure. Sensors are typically connected in an ad-hoc way. In this project, we will focus on localization approaches which are ad-hoc, non-map based and active.

Our research aims to design effective RF-based localization solutions for wireless sensor networks. There are many different types of wireless sensor networks. It is impractical to completely re-design a localization method for each type of sensor network. We will identify the common features shared by all wireless sensor networks, and propose some generic techniques which apply well to all of the types. Sensor networks can be classified according to the availability of anchors and the mobility of the sensors. In this research, three representative types of sensor networks will be selected for careful study. Based on the unique characteristics of each type of sensor network, different approaches will be proposed. Our research will address the localization problem for the following three types of wireless sensor network.

- Type 1: *Static sensor networks without anchors.*

An anchor is a sensor node which has prior knowledge of its physical absolute location, via GPS or manual configuration. In this type of sensor network, after deployment, the sensor nodes are stationary. No anchors are required, which helps reduce the deployment cost and increase the flexibility of the sensor network. Since there is no a global coordinate system for reference, the goal is to compute the relative location graph of sensors.

• Type 2: *Static sensor networks with anchors.*

In this type of sensor network, a number of anchors are deployed before localization. The anchors are equipped with accurate location information, and the localization is expected to reference to the accurate locations of the anchors. Since there is a global coordinate system for reference, the goal is to compute absolute locations of sensors in this coordinate system.

• Type 3: *Mobile sensor networks.*

In mobile sensor networks, there are some mobile sensors which are allowed to move freely. These mobile sensors are different from stationary sensors in that it is very important for each mobile sensor to update its location frequently, so that its location information remains meaningful. Mobile senor networks are very useful in some situations, e.g., when robots are involved.

In brief, our approaches include two novel techniques: Quality of localization (QoL) and Refinements.

1) Quality of localization (QoL). Localization errors are inherently associated with every computed location. Since the goal is to compute locations for sensor nodes as accurately as possible, these localization errors should be carefully taken into account throughout the process of the localization. The key question is how localization errors can be considered during the localization process to help increase the localization accuracy. We will introduce the concept of Quality of Localization (QoL) to indicate the accuracy of a computed location for a specific sensor node. Any computed location is associated with a QoL. A better QoL means that the computed location is much closer to the real location (less localization error). There are many factors affecting the QoL, such as relative position and number of reference nodes. The basic idea is that the QoL of any newly computed location should be determined based on the QoLs of the locations of those referenced nodes. With the knowledge provided by QoL, the localization can hopefully be enhanced in terms of localization accuracy.

2) Refinements. Although the location of a sensor node can be uniquely computed by reference to three other sensor nodes, we believe that the accuracy of location can be significantly improved if we can make use of as much information from other reference nodes as possible. To this end, we will propose a refinement scheme. The basic idea is that the location of a sensor node may be computed independently by its neighbors, and the location of a sensor node can be refined by taking into account all its computed locations.

The success of this research will lead to effective and inexpensive RF-based localization techniques, which will significantly contribute to the development of sensor networks.

2.3 CABOT: Ontology-Based Middleware

2.3.1 Cabot Overview

In pervasive computing, context is defined as any information that can be used to characterize the situation of an entity (a person, place, or object) that is considered relevant to the interaction between a user and an application, including the user and application themselves[26]. A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task[26].

There is growing interests in the use of context-awareness as a technique for developing pervasive computing applications that are flexible, adaptable, and capable of acting autonomously on behalf of users[27]. Up to now, promising context-aware techniques have been adopted in many applications like proximate selection (items relevant to the user's contexts are emphasized), automatic contextual reconfiguration (automatic binding to available resources based on current contexts), contextual commands (executable services are made available based on the user's contexts) and context-triggered actions (services are executed automatically when the right context exists)[28].

However, context-awareness imposes a significant requirement for reliable capturing and efficient management of context data, which introduces various challenges concerning consistency, reliability, and usability in the data capturing and management of sensor network applications. This motivates our development of a context management middleware, called Cabot, to enable the deployment of large-scale, reliable and reusable context-aware applications over sensor networks. In particular, reliable data capturing and analysis, consistency management and repairing, context-aware exception handling are key functions of Cabot.

The research on pervasive data capturing and management support has a close relation to context data modeling and abstraction technologies. The pioneering work by Schilit *et al.*[28] uses dynamic environment servers to manage and disseminate context data. Later work further studied models of context data processing[29,30], context-sensitive data structures[31], and other issues. These technologies model pervasive context data, but only gave preliminary considerations for data reliability and consistency. Advanced issues about inconsistency detection and data repairing are left untouched.

Previous work has also tackled several challenging problems in context data processing, reasoning and programming support[27,32−36], but little consideration has been given to an integrated support of exception

handling and context management under a publish-subscribe framework.

Previous work has also tackled several challenging problems in context data processing, reasoning and programming support[27,32−35], but little consideration has been given to the support of exception handling in context data processing and management.

Some research projects on pervasive resource management and collaboration, including EasyLiving[37], Gaia[38], i-Land[39], Interactive Workspaces[40], MobiPADS[41] and RCSM[42], have been proposed to provide the middleware support for mobile or pervasive computing. These projects are mainly concerned about the organization of and the collaboration amongst pervasive computing resources (including devices and services), but most of them requires special sensing environments. Amongst them, Gaia behaves better by focusing on general purpose environments, but it has not tackled some important issues like the adaptation to various sensor capabilities and context-aware exception handling.

2.3.2 Cabot System Architecture

To address the technical challenges in the context management at the middleware layer, we decompose the context management of Cabot into three layers, namely raw context collection, context processing and context application, as depicted in Fig.6.

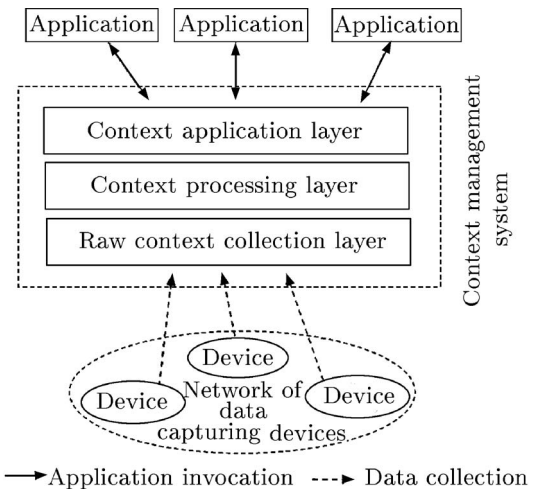


Fig.6. Architecture of the context management system.

The Raw Context Collection Layer abstracts the underlying data capturing networks, decoupling the upper layers from underlying devices and platforms. The main function in this layer is to deal with the issues arising from different device models. For example, existing sensor models support a variety of control modes, data transferring speeds and data formats. The Context Collection Layer not only integrates different kinds of data capturing networks, but also provides high degree of extensibility, allowing new networks to be readily integrated. To address the decoupling issue, we abstract data capturing devices by means of widgets, which accept and store raw data from these devices. Context sensing of these devices can thus be supported through predefined widget interfaces. To achieve high degree of extensibility, the widget interfaces are such designed to facilitate the deployment of a mediator-observer pattern, which facilitates the addition of context sensing sources (i.e., widgets) and sinks (i.e., services consuming contexts at higher layers). However, this only partially solves the heterogeneity problem. When new data capturing devices are introduced, the associated contexts may be subject to new meanings. As such, we augment the context sensing mechanism using an ontology, which helps resolve the semantic differences in sensed contexts.

The Context Processing Layer provides transparent access for context data and decouples physical context sources from applications. Context sources can be referenced logically. This layer supports the context presentation management. Based on the raw context data acquired from the Raw Context Collection Layer, new contexts are generated. One issue in context presentation is its expressiveness. How to design a context model adequate for most applications is a major challenge. Moreover, context may be generated by combining multiple raw context data from different sources. To address this problem, a context is represented in terms of a 4-ary tuple (*Category, Fact, Constraints, Attributes*). Category describes the type of the context, which can refer to location, situation, move, and so on. *Fact* = (*Subject, Predicate, Object*) gives the content of the context, where *Subject* and *Object* are related by a Predicate. *Constraints* = (*Lifespan, Site*) specifies two constraints relevant to the context, where *Lifespan* and *Site* refer to temporal and spatial constraints, respectively. Examples of Lifespan include "at 14:30pm on Jan. 1, 1999" or "from Jun. 9, 2004 to Jul. 11, 2004", and *Site* examples include "City Hall" or "HKUST campus". *Attributes* = (*Nature, Precision*) describe two properties of the context. *Nature* indicates whether the context is a sensed context, derived context, profiled context or static context. Sensed contexts (e.g., Peter comes into the surgery room) are collected by data capturing devices. Derived contexts (e.g., Michael becomes unconscious) are derived from other contexts based on predefined rules. Profiled contexts (e.g., Peter takes care of Michael) and static contexts (e.g., Michael was born in Jan. 1977) are supplied by human operators. Static contexts remain unchanged across time, while the other three are dynamic — sensed contexts typically change frequently; derived contexts and profiled contexts keep evolving in a medial speed amongst the four. *Precision* is a percentage value specifying the accuracy of the context. The value can be provided by hardware engineers, human operators or mechanically derived.

The Context Application Layer provides applica-

tions with more sophisticated context awareness supports such as context consistency management and exception handling. A key requirement in context consistency management is the ability to bridge the gap between: (i) contexts identified by the Context Processing Layer, (ii) situations under which an application needs to react, and (iii) repairing actions that should be taken. As such, the consistency management module provides at least three services: (i) offering necessary reasoning capabilities in inconsistency detection; (ii) defining application specific inconsistency detection rules; and (iii) realizing an automatic inconsistency repairing mechanism. These services are driven by the following two reasoning mechanisms.

1) *General Reasoning.* Various services and applications may have some common requirements of concept semantics reasoning in inconsistency detection. For example, "a person enters the surgery room" and "a human being comes into the surgery room" are commonly regarded as having the same meaning despite the notation difference ("person" vs. "human being", and "enter" vs. "come into") by various context sources. Another example is that: assuming that a more general concept can cover a more specific concept, users may hope to use "vehicle" to refer to the semantics of "vehicle" and all its sub-concepts (e.g., "bus" and "car") so that they can capture richer contexts.

2) *Application Specific Reasoning.* Different services and applications share few common rules for inconsistency detection. Some contexts cause inconsistencies in one scenario, but may be acceptable in another scenario. With the knowledge that Peter is not allowed to enter a confidential zone Z, a problematic context, "Peter appears in Z", could be a result of incorrect monitoring of Z in a scenario. However, the context might be acceptable in another scenario when Peter is accompanied by an authorized person.

The Context Application Layer also provides exception handling mechanism to applications when context inconsistency occurs. Such a mechanism can also be context-aware. The exceptional handling module keeps listening to the context consistency management module. When inconsistency occurs, the module fires predefined exception handlers which can take appropriate actions to compensate the current application activities and/or trigger repairing actions at the problematic context sources. At present, Cabot is still at a prototyping stage. New functionalities and features (e.g., context trigger and context deriving) will be incorporated into the future releases of Cabot.

2.4 MEADOWS: A Suite of Evaluation Tools

We are developing a collection of tools for large-scale, in-depth studies on sensor networks. Specifically, these tools are on modeling, emulation, and data analysis for wireless sensor networks (MEADOWS)[43] in addition to monitoring. We have designed a hierarchical power consumption model for sensor databases, a distributed emulator called VMNet (Virtual Mote Network) of sensor networks[44], and a few data analysis functions for sensory data. These tools will interact with all layers of BLOSSOMS closely. On the one hand, information about real systems (e.g., sensor node statistics from the BUDS layer) provides system parameters for MEADOWS; on the other hand, MEADOWS enables analysis and validation of real systems. In this section, we focus on the VMNet sensor network emulator.

2.4.1 VMNet Overview

In-depth studies on wireless sensor networks (WSNs) require realistic simulation and emulation environments. As the resources (computation, communication, and power) in current WSNs are limited, it is more feasible to perform simulation studies on PCs than instrumenting the real sensor nodes for performance evaluation. Moreover, because real WSNs are cost-expensive, hard to maintain, and tightly embedded in the physical world, it is desirable to conduct repeatable experiments in a realistic but controllable environment. As evidence, recent work (e.g., TOSSIM[45] and EmStar[46]) has provided such software environments for the initial development and deployment of WSN applications. In this paper, we follow up with VMNet, a WSN emulator that we built to enable realistic performance evaluation of WSN applications.

The major difference between VMNet and the existing WSN simulation tools such as EmStar and TOSSIM is that VMNet is a general WSN emulator whereas both EmStar and TOSSIM are TinyOS-specific simulators. VMNet has an open architecture for instruction-level sensor node CPU emulation and signal-level radio channel emulation. The application code that is compiled for a specific type of sensor nodes can be run directly with the emulation modules for the type of sensor nodes in VMNet. In contrast, EmStar and TOSSIM are fixed with the TinyOS virtual processor (not the CPU or radio channel hardware emulator) and the application code must becompiled for the PC-version TinyOS to be run in the simulators. Moreover, we have developed performance models on response time and power consumption with parameter values from real-world measurements. As a result, VMNet is able to evaluate application performance more accurately than previous tools. Finally, VMNet allows us to plug in emulation modules for new types of sensor nodes, such as the ones we are developing, and to evaluate the application performance of these nodes.

We choose emulation over simulation in VMNet for fidelity and user convenience. As newly emerged, networked and embedded systems, WSNs mix various known or unknown performance issues in a single instance. Therefore, we start from detailed emulation and

gradually raise the level of abstraction rather than abstracting a few performance factors in the very beginning. Furthermore, emulation is convenient to users as binary code compiled for a target system can be executed directly in an emulator. This feature also eliminates possible interference from implementation and compilation. Additionally, emulation of the entire system allows the evaluation of cross-layer techniques for OSs, protocols and applications of WSNs.

In VMNet, a target WSN is emulated as a VMN (Virtual Mote Network). Each sensor node (or called mote) is emulated as a VM (Virtual Mote). The CPU of a mote is emulated at the clock cycle level, and the sensing units and other hardware peripherals are also emulated in sufficient detail. The radio signal transmission is emulated by the communication between VMs with the effects of signal loss and noise. Finally, the binary code of the target WSN can be run directly on the VMN for debugging, testing and, most importantly, performance evaluation.

We select two performance metrics for evaluation: one is response time, and the other is power consumption. Response time is critical to some WSN applications, such as chemical detection, whereas power consumption is the major concern for applications in which replacing batteries is inconvenient or infeasible. In addition, the longer the response (working) time, the greater the power consumption, if other factors (e.g., voltage) are fixed.

In order to evaluate the response time and power consumption, VMNet logs detailed running status of the application code under evaluation. The running status is categorized by the operations of the CPU, the sensing units, and the communication units. The virtual running time (i.e., the time in the emulated world) of each operation is also logged. Using these logged statistics from emulation and the parameter values (e.g., electric current) from real world measurement, VMNet is able to calculate the two performance metrics.

We have evaluated VMNet in comparison with real target WSNs and TOSSIM. Our results demonstrate that, with the measurement statistics of the real WSNs and the running status logged in VMNet, the estimated running time and power consumption in VMNet are close to those in the target WSNs. In addition, VMNet facilitates detailed study on running status and data delivery schemes. Due to its high fidelity, VMNet has an emulation speed of 1/10 of that of TOSSIM, but its scalability curve with respect to the number of motes emulated is similar to that of TOSSIM.

2.4.2 VMNet Architecture and Components

Similar to a target WSN, VMNet has a user client, a proxy, and virtual motes in the VMN (see Fig.7). VM_0 is the virtual sink node, and it communicates with other VMs via the virtual radio channel. Since the user client and the data proxy are PC programs, they are the same as those in the target WSN except the connection between the data proxy and the sink node (e.g., the RS232 serial connection), which is emulated in VMNet. With this emulation architecture, a VMN works the same way as its target WSN. Users submit queries at the user client, and the queries are parsed and disseminated through the proxy to the virtual motes. The application code is executed at the virtual motes, and the sensory data (query results) are returned from the virtual motes through the sink node and the proxy to the user client.

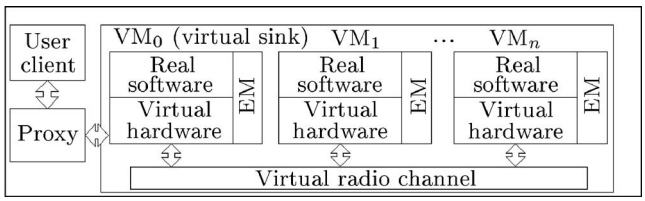


Fig.7. VMNet system architecture.

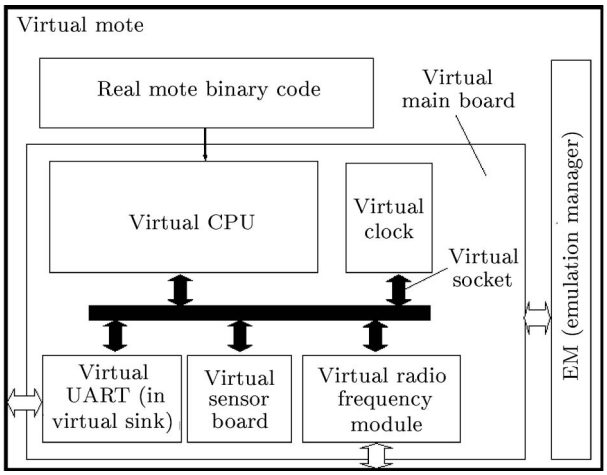


Fig.8. Virtual mote (VM).

As shown in Fig.8, a VM contains both virtual hardware and real software components. The real software component is the binary code that is compiled for the real motes, which handles hardware interrupts, MAC (medium access control) layer communication, and query processing. The virtual hardware components include a Virtual CPU (VCPU), a virtual clock, a virtual sensor board, and a Virtual Radio Frequency Module (VRFM). If the VM is a sink node, it has a virtual UART (universal asynchronous receiver/transmitter) to emulate the communication interface between the sink and a proxy (PC). Finally, all virtual hardware components are contained in the virtual main board, which provides a number of virtual sockets to connect the virtual hardware components. Additionally, the EM (Emulation Manager) in each VM controls the VM operations and logs the running sta-

tus. These running status statistics enable debugging and performance evaluation. With these components, a VM performs tasks of a real mote. It acquires sensory data, receives and transmits bits via radio channels, and executes CPU instructions and handles interrupts.

For realistic performance evaluation in VMNet, we currently adopt the hardware configuration of the widely used Crossbow[2] MICA2 motes. This configuration includes the Atmel[47] Atmega128 CPU and the Chipcon CC1000 radio frequency circuit (main part in the RFM). For this configuration, we modified the CPU emulator in Atemu 4.0[48] as a CPU emulation module in a VM, and we implemented our own VRFM to emulate the MICA2 RFM. The VRFM performs the coding/decoding (e.g., Manchester and Non-Return-to-Zero) and virtual radio signal transmitting/receiving functions as the real RFM. We use Bullington's general radio path loss model[49] to simulate the attenuation of radio signals. To simulate the noise, we use a Gaussian model to compute the BER (Bit Error Rate) from SNR (Signal-to-Noise Ratio) in VMNet.

2.4.3 VMNet Evaluation

For evaluation, we first measured electric currents in small-scale target WSNs using electric equipment. We then compared the performance of applications on 2-node, 5-node and 10-node networks evaluated in TOSSIM, VMNet and real WSNs to study VMNet's accuracy. Finally, we evaluated the scalability of VMNet in comparison with TOSSIM.

Table 1 lists the equipment used in the experiments. The software used in these experiments was TinyOS[6] (V1.1.0), TinyDB[50] (V0.2) and TOSSIM. We ran VMNet or TOSSIM with the user client and the proxy in TinyDB in one PC (single-PC emulation/simulation).

Table 1. Equipment Used in VMNet Evaluation

Name	Model	Quantity
Oscillscope	HP 4155A	1
PC	Pentium® 3.2GHz, 1GB RAM, 40GB hard disk, running Red hat Linux 9.0	1
Sink mote	MIB510 interface board + MPR 410 processor board	1
Sensor mote	MICA2 (MPR 410 processor board + MTS300 sensor board)	9

We used the oscilloscope in an electronic lab to measure the electric current and the voltage of a sensor mote. With the electric current and voltage of a sensor mote measured, its power consumption can be computed by the following equation:

$$W = \sum_{i=0}^{N} U_i I_i t_i,$$

where U_i and I_i are the voltage and the electric current in a measurement, N the total number of measurements, and t_i the interval between two measurements.

We deployed three WSNs with 2, 5, and 10 nodes each. In each WSN, every node was placed within 4 m^2 around the sink node. The configurations of the emulated WSNs (VMNs) in VMNet and in TOSSIM were the same as those of the target WSNs we measured. The application used was a TinyDB query (shown as Query 1). For each query execution, we recorded the time and energy cost of one mote returning two query results of Query 1. In total, we recorded the time and energy cost of the mote for 30 times and computed the average in TOSSIM, VMNet and in the real WSNs.

Query 1. *Sample the temperature every* 2,048 *ms.*

Table 2. Comparison of VMNet, TOSSIM, and Target WSNs

Metrics	Power consumption (Joules)		Time (ms)		
network					
#nodes	WSN	VMNet	WSN	VMNet	TOSSIM
2	0.229	0.213	3,781.8	4,043.3	3,741.9
5	0.268	0.225	4,214.2	4,186.2	3,729.6
10	0.303	0.260	4,426.5	4,585.2	3,725.1

The performance results are shown in Table 2. As in Table 2, when the number of nodes increases, the query response time in the real WSNs increases from 3.8 seconds to 4.4 seconds. This increase indicates that the size of network affects the query response time. One possible cause is that with a higher density of the nodes (more nodes occupying the same size of area), the collision of radio signals is more severe and it therefore takes more time to finish the transmission. In comparison, the query response time estimated in VMNet increases from 4.0 seconds to 4.6 seconds whereas that estimated by TOSSIM remains at around 3.7 seconds, when the number of node increases from 2 to 10. Additionally, both the real WSNs and VMNet report increasing power consumption when the number of nodes increases.

Given the high fidelity of VMNet, the emulation speed, or the scalability of VMNet is a concern to address. Therefore, we studied the scalability of VMNet with that of TOSSIM[48], which is known to be highly scalable, on a single PC. We chose TinyDB, as opposed to other simple applications, for the scalability test, because we believe that query processors for sensory data acquisition are one of the major WSN applications.

In our experiments, we fixed the density of the emulated network to be one node per nine square meters and varied the total number of emulated nodes from 10 to 100. We ran Query 1 (specified in Subsection 4.3) and recorded the emulation time (not the virtual time) of emulating 10 seconds virtual time (7.3728×107 VCPU clock cycles) in VMNet. We also measured the simulation time of TOSSIM under the same configuration as that of VMNet. We ran both VMNet and TOSSIM on a single PC (configuration specified in Table 1) and compared their scalability.

Fig.9 shows the time needed by VMNet and TOSSIM to complete the emulation/simulation of the 10 virtual

seconds of query processing. The speed of VMNet was about 1/10 of that of TOSSIM, because VMNet emulates sensor nodes at the instruction level of a sensor mote and logs detailed running state statistics. This is the tradeoff between speed and fidelity.

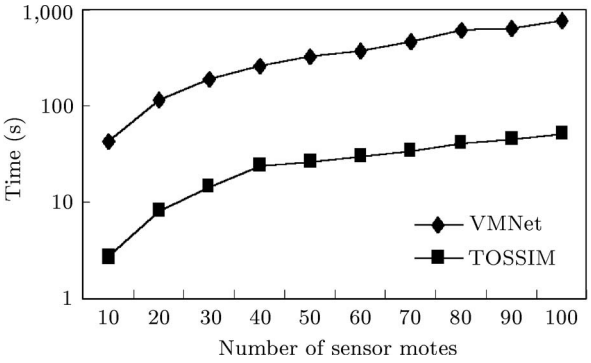


Fig.9. Scalability of VMNet and TOSSIM.

Interestingly, the shape of the scalability curve of VMNet was almost identical to that of TOSSIM. The main reason was that both VMNet and TOSSIM ran the same application (Query 1 on TinyDB) with the same OS (TinyOS), and the processing flows in VMNet and in TOSSIM were similar. Additionally, TOSSIM was shown to be computation-intensive[48]. This CPU bottleneck was also true for single-PC VMNet, and we are extending VMNet to run on multiple networked PCs to alleviate this bottleneck.

3 Conclusions

We have presented an overview of the BLOSSOMS project being investigated at CAS and HKUST. Building working sensor nodes (BUDS) has laid a solid foundation for the project. The routing protocols and localization mechanisms in sensor networks facilitate upper-level software to communicate data and commands reliably. The CABOT middleware provides infrastructural support for applications and the MEADOWS tools allow controllable, realistic studies of large-scale sensor applications. This project involves close collaboration of researchers from different disciplines because of the broad project scope. At the time of writing, the project is still in its early stage. As the project moves forward, we expect to identify more research issues and produce more research results.

References

[1] Akyildiz I, Su W, Sankarasubramaniam Y *et al.* A survey on sensor networks. *IEEE Communications,* August 2002, pp.102–114.

[2] Crossbow, Inc. Wireless sensor network products. http://www.xbow.com.

[3] Hill J, Szewczyk R, Woo A *et al.* System architecture directions for network sensors. *Computer Achitectural Support for Programming Languages Operating Systems* 2000, Cambridge, MA, pp.93–104.

[4] Gao W, Ni L M, Xu Z W. BLOSSOMS: A CAS/HKUST joint project to build lightweight optimized sensor systems on a massive scale. In *Proc. the IFIP NPC'04 Workshop on Building Intelligent Sensor Networks* (*BISON'04*), Oct. 2004.

[5] The Smart Dust project. http://robotics.eecs.berkeley.edu/~pister/SmartDust/

[6] TinyOS. http://www.tinyos.net

[7] Arici T, Altunbasak Y. Adaptive sensing for environment monitoring using wireless sensor networks. In *Proc. IEEE Wireless Communications and Networking Conference* (*WCNC*), Atlanta, GA, 2004, pp.2347–2352.

[8] Delin K A, Jackson S P. The sensor web: A new instrument concept. *SPIE's Symposium on Integrated Optics,* San Jose, CA, Jan 2001, pp.1–9.

[9] Korhonen J P, Van Gils M. Health monitoring in the home of the future. *IEEE Engineering in Medicine and Biology Magazine,* May 2003, pp.66–73.

[10] Mainwaring A, Culler D, Polastre J, Szewczyk R, Anderson J. Wireless sensor networks for Habitat monitoring. In *Proc. 1st ACM International Workshop on Wireless Sensor Networks and Applications,* Atlanta, Georgia, 2002, pp.88–97.

[11] Schwiebert L, Gupta S K S, Weinmann J. Research challenges in wireless networks of biomedical sensors. *Mobile Computing and Networking,* 2001, pp.151–165.

[12] Braginsky D, Estrin D. Rumor routing algorithm for sensor networks. In *Proc. 1st International Workshop on Wireless Sensor Networks and Applications* (*WSNA'02*), Atlanta, Georgia, 2002, pp.22–31.

[13] Ganesan D, Govindan R, Shenker S, Estrin D. Highly-resilient, energy-efficient multipath routing in wireless sensor networks. In *2nd ACM International Symposium on Mobile Ad Hoc Networking* (*MobiHOC 2001*), Long Beach, CA, 2001, pp.251–254.

[14] He T, Stankovic J A, Lu C *et al.* SPEED: A stateless protocol for real-time communication in sensor networks. *IEEE International Conference on Distributed Computing Systems* (*IEEE ICDCS*), Providence, RI, 2003, pp.46–55.

[15] Heinzelman W R, Chandrakasan A, Balakrishnan H. LEACH: Energy-efficient communication protocol for wireless microsensor networks. *Hawaii Internation Conference on System Sciences,* 2000.

[16] Intanagonwiwat C, Govindan R, Estrin D. Directed diffusion. *International Conference on Mobile Computing and Networking* (*MobiCom 2000*), Boston, USA, 2000, pp.56–67.

[17] Ko Y B, Vaidya N H *et al.* Location-aided routing (LAR) in mobile ad hoc networks. In *Proc. the ACM/IEEE International Conference on Mobile Computing and Networking* (*MobiCom98*), Dallas, USA, 1998.

[18] Mauve M, Widmer J, Hartenstein H. A survey on position based routing in mobile ad-hoc networks. *IEEE Network Magazine,* 2001, 15: 30–39.

[19] Xu Y, Heidemann J, Estrin D. Geography-informed energy conservation for ad hoc routing. *International Conference on Mobile Computing and Networking* (*MobiCom'01*), Rome, Italy, 2001, pp.70–84.

[20] Doherty L, Pister K, Ghaoui L E. Convex position estimation in wireless sensor networks. *IEEE Infocom,* Anchorage, AK, 2001, pp.1655–1663.

[21] He T, Huang C, Blum B M *et al.* Range-free localization schemes for large scale sensor networks. *International Conference on Mobile Computing and Networking* (*MobiCom'03*), San Diego, California, 2003, pp.81–95.

[22] Iyengar R, Sikdar B. Scalable and distributed GPS free positioning for sensor networks. *International Conference on communication* (*ICC*), Anchorage, Alaska, 2003, pp.338–342.

[23] Langendoen K, Reijers N. Distributed localization in wireless sensor networks: A quantitative comparison. *Computer Networks, Special Issue: Wireless Sensor Networks,* 2003, 43:

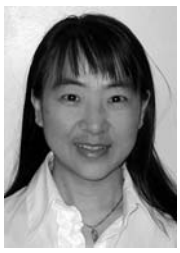
499–518.
[24] Ni L M, Liu Y H, Lau Y C *et al.* LANDMARC: Indoor location sensing using active RFID. In *First IEEE Int. Conf. Pervasive Computing and Communications* (*PerCom'03*), Fort Worth, Texas, 2003, pp.407-415.
[25] Priyantha N B, Chakraborty A, Balakrishnan H. The cricket location-support system. *International Conference on Mobile Computing and Networking* (*MobiCom2000*), Boston, MA, 2000, pp.32–43.
[26] Anind K Dey, Gregory D Abowd. Towards a better understanding of context and context-awareness. VU Technical Report, GIT-GVU-99-22, 1999.
[27] Karen Henricksen, Jadwiga Indulska. A software engineering framework for context-aware pervasive computing. In *Proc. the 2nd IEEE Annual Conference on Pervasive Computing and Communications* (*PerCom 2004*), Orlando, Florida, USA, Mar. 2004.
[28] Bill N Schilit, Marvin M Theimer, Brent B Welch. Customizing mobile applications. In *Proc. USENIX Mobile & Location-Independent Computing Symposium,* Cambridge, Massachusetts, USA, Aug. 1993, pp.129–138.
[29] Andy Harter, Andy Hopper, Pete Steggles *et al.* The anatomy of a context-aware application. *Mobile Computing and Networking,* 1999, pp.59–68.
[30] Albrecht Schmidt, Kofi Asante Aidoo, Antti Takaluoma *et al.* Advanced interaction in context. In *Proc. the 1st International Symposium on Handheld and Ubiquitous Computing* (*HUC 1999*), Karlsruhe, Germany, Sep. 1999, pp.89–101.
[31] Payton J, Roman G-C, Julien C. Context-sensitive data structures supporting software development in ad hoc mobile settings. In *Proc. the 3rd International Workshop on Software Engineering for Large-Scale Multi-Agent Systems* (*SELMAS 2004*), Edinburgh, Scotland, UK, May 2004.
[32] William G Griswold, Robert Boyer, Steven W Brown, Tan Minh Truong. A component architecture for an extensible, highly integrated context-aware computing infrastructure. In *Proc. the 25th International Conference on Software Engineering* (*ICSE 2003*), Portland, Oregon, USA, May 2003.
[33] Hisazumi K, Nakanishi T, Kitasuka T, Fukuda A. A context-aware middleware mapping processes and user-context subspaces. In *Proc. the International Conference on Parallel and Distributed Processing Techniques and Applications* (*PDPTA 2003*), Las Vegas, NV, USA, Jun. 2003.
[34] Ranganathan A, Al-Muhtadi J, Campbell R H. Reasoning about uncertain contexts in pervasive computing environments. *IEEE Pervasive Computing,* 2004, 3(2): 62–70.
[35] Chang Xu, S C Cheung, Cindy Lo, K C Leung, Jun Wei. Cabot: On the ontology for the middleware support of context-aware pervasive applications. In *Proc. the IFIP NPC'04 Workshop on Building Intelligent Sensor Networks* (*BISON'04*), Wuhan, China, October 2004, pp.568–575.
[36] Considine J, Li F, Kollios G, Byers J. Approximate aggregation techniques for sensor databases. In *Proc. the 20th International Conference on Data Engineering,* 2004, pp.449–460.
[37] Brumitt B, Meyers B, Krumm J, Kern A, Shafer S. EasyLiving: Technologies for intelligent environments. In *Proc. the 2nd International Symposium on Handheld and Ubiquitous Computing* (HUC 2000), Bristol, England, 2000.
[38] Roman M, Hess C, Cerqueira R *et al.* A Middleware infrastructure for active spaces. *IEEE Pervasive Computing,* 2002, 1(4): 74–83.
[39] Peter Tandler. Software infrastructure for ubiquitous computing environments: Supporting synchronous collaboration with heterogeneous devices. In *Proc. the 2nd Int. Conf. Ubiquitous Computing* (UbiComp 2001), Atlanta, Georgia, USA, 2001.
[40] Fox A, Johanson B, Hanrahan P, Winograd T. Integrating information appliances into an interactive workspace. *IEEE Computer Graphics and Applications,* 2000, 20(3): 54–65.
[41] Alvin T S Chan, Siu-Nam Chuang. MobiPADS: A reflective middleware for context-aware mobile computing. *IEEE Trans. Software Engineering,* 2003, 29(12): 1072–1085.
[42] Yau S S, Karim F. A context-sensitive middleware for dynamic integration of mobile devices with network infrastructures. *J. Parallel & Distributed Computing,* 2004, 64(2): 301–317.
[43] Luo Q, Ni L M, He B, Wu H, Xue W. MEADOWS: Modeling, emulation, analysis of data of wireless sensors. In *Proc. the VLDB Workshop on Data Management for Sensor Networks* (*DMSN'04*), Toronto, Canada, August 2004.
[44] Wu H, Luo Q, Zheng P, He B, Ni L M. Accurate emulation of wireless sensor networks. In *Proc. the IFIP NPC'04 Workshop on Building Intelligent Sensor Networks* (*BISON'04*), Wuhan, China, October 2004, pp.576–583.
[45] Philip Levis, Nelson Lee, Matt Welsh *et al.* TOSSIM: Accurate and scalable simulation of entire TinyOS applications. In *Proc. the First Int. Conf. Embedded Networked Sensor Systems,* Sensys 2003, Los Angeles, 2003.
[46] Lewis Girod, Jeremy Elson, Alberto Cerpa *et al.* EmStar: A software environment for developing and deploying wireless sensor networks. USENIX'04, Boston, MA, 2004.
[47] Atmel. http://www.atmel.com/products/avr/.
[48] Atemu. http://www.cshcn.umd.edu/research/atemu/.
[49] Bullington K. Radio propagation for vehicular communications. *IEEE Trans. Vehicular Tech.,* Nov. 1977, VT-26(4): 295–308.
[50] TinyDB. http://telegraph.cs.berkeley.edu/tinydb/.

Wen Gao received his two Ph.D. degrees in computer science from Harbin Institute of Technology, China, in electronics engineering from the University of Tokyo, in 1988 and 1991 respectively. He served as the chairman of steering committee for intelligent computing system in 863 Hi-Tech programme from 1996 to 2001. He is the head of Chinese delegation to MPEG. He is also the chair of AVS working group which is an entity to make and evaluate the national standard for audio/video coding system. His research interests include application of artificial intelligence, multimiedia, data compression, face recognition, sign language recognition and synthesis, image retrieval, and multimodal interface.

Lionel M. Ni earned his Ph.D. degree in electrical and computer engineering from Purdue University, West Lafayette, IN, in 1980. He is professor and Head of Computer Science Department of Hong Kong University of Science and Technology. His research interests include parallel architectures, distributed systems, high-speed networks, and pervasive computing. He is a fellow of IEEE. Dr. Ni has chaired many professional conferences and has received a number of awards for authoring outstanding papers. His paper (with his former student, Chris Glass) "The Turn Model for Adaptive Routing" was selected as one of the 41 most significant impact papers in the last 25 years in computer architecture area in 1998. He also won the Michigan State University Distinguished Faculty Award in 1994.

Zhi-Wei Xu received his Ph.D. degree in computer engineering from University of Southern California in 1987. He is professor and Vice Director of ICT, CAS. He serves on the Editorial Board of *International Journal of Grid Computing*. His research interests include grid computing, parallel architectures and pervasive computing.

S. C. Cheung received his B.Eng. (Hons.) degree in the electrical engineering from the University of Hong Kong in 1984. He received the M.Sc. and Ph.D. degrees in computing from the Imperial College of Science, Technology and Medicine, University of London, London, U.K in 1988 and 1994, respectively. He is an associate professor of computer science and associate director of CyberSpace Center at the Hong Kong University of Science and Technology. He has served actively on the program committees of international conferences on software engineering, distributed systems and web technologies. His research interests include software engineering, sensor networks, pervasive computing, services computing and information security.



Li Cui received the B.S. degree from Tsinghua University, China, in 1985 and the M.Sc. degree from the Institute of Semiconductors, Chinese Academy of Sciences, in 1988, where she worked on solid state and electrochemical sensors. She received the Ph.D. degree in electronic nose application from the University of Glasgow, UK, in 1999. She is currently a professor at the Institute of Computing Technology, Chinese Academy of Sciences. Her research interests include sensor technology and sensor networks.

Qiong Luo is an assistant professor at the Department of Computer Science, the Hong Kong University of Science and Technology (HKUST). She received her Ph.D. degree in computer sciences from the University of Wisconsin-Madison, USA, in 2002. Her research interests are database systems, with a focus on data management and analysis techniques related to network applications. Her current research is focused on query processing in pervasive computing and sensor networks.