Mobile Agent based Data Fusion for Wireless Sensor Networks with a XML Framework

Gehao Lu*[,1,2], Joan Lu[1] and Shaowen Yao[2]

[1]*Department of Computing and Engineering, University of Huddersfield, UK*

[2]*Laboratory of Network Intelligent Computing, Yunnan University, P.R. China*

Abstract: In Wireless Sensor Network, data fusion or data aggregation is the key technology to decreased power consumption and improved accuracy of data collection. This paper proposes a new data fusion approach which is based on the mobile agent technology. It uses mobile agent communication to decide the filtering or merging of data. It also makes use of XML as the expressive mechanism for sensed data, network topology, and agent's inner framework. An experiment is designed to evaluate the proposed approach with the existing data fusion technology. It is found that the new approach exhibits better performance in data fusion efficiency and power consumption savings.

INTRODUCTION

The fundamental functionality of wireless sensor network is to collect and return data from the sensor nodes [1]. Data fusion plays an important role because it can save communication bandwidth and power while improving the efficiency of data collection [2]. The popular data fusion methods for wireless networks include address-centric routing [3] and data-centric routing [4]. In address-centric routing, each source node will forward its data along the shortest path while not considering the routing of data fusion; in the data-centric routing, the node will analyze the content of the data and process the data with filtering or merging operation [5]. These current solutions for data fusion exhibit the following disadvantages [6]:

Incorporating data fusion operations among nodes introduces significant delay, thus sacrificing the efficiency of the whole network.

There is no mature solution in the application layer which fully facilitates the distributed database technology.

The middle layer nodes cannot communicate with each other. There is no sharing of information between different branches of the sensor network.

This article proposes a new solution which makes use of the mobile agent to traverse the wireless sensor network. The architecture of the solution is illustrated in Fig. (**1**). While traversing the nodes, the agent collects the sensed data and node information . All the information including the inner framework of agents is described in the XML form. The article also illustrates the XML schema use to confine the XML files. The data fusion is carried out by two agents meeting in a node. The merging process is a communication process between two agents. It is also a procedure that makes use of XML processing. The researcher also designs experiments to evaluate this new approach. The result is depicted in the form of comparison between traditional data fusion approach and the proposed new approach.

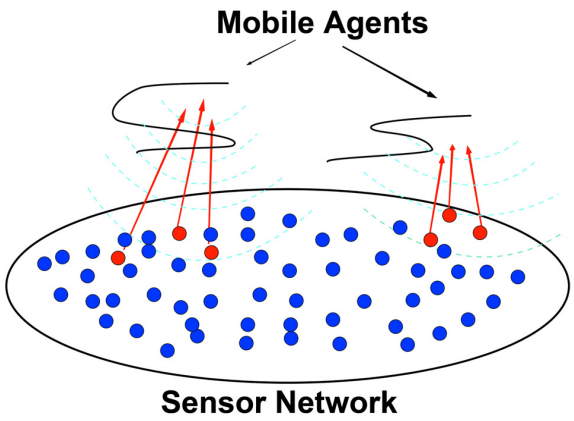


Fig. (1). Mobile agent based sensor network [7].

Section 2 provides the technical background which introduces the state of art in the field of data fusion for wireless sensor networks. This section also discusses the advantages and disadvantages of those well-known fusion methods. Section 3 introduces the methods and algorithms employed in the proposed mobile agent based solution. It describes the agent dispatching and returning algorithm in details. Section 4 introduces the design of XML Schema which is used to depict the inner structure of mobile agent and the data fusion related information elements. Section 5 compares the new approach with the traditional data fusion technique. It uses the form of comparison picture to evaluate the effectiveness of the new approach. Section 6 discusses some potential problems and its solution in the newly proposed method. Section 7 is the conclusion and future work.

TECHNICAL BACKGROUND

The distributed database technology is widely used in the process of collecting data in sensor networks. Researchers

*Address correspondence to this author at the Department of Computing and Engineering, University of Huddersfield, UK; E-mail: j.lu@hud.ac.uk

1874-1339/09

exploit languages similar to SQL (Structured Query Language) to aggregate data in the application layer of the sensor networks [8]. The queries with requests of data collecting are sent to all the nodes in the network. Each upper node processes data from its lower nodes and decide what data to be sent to its upper node. The advantages of such mechanism include: it hides the details of implementation under the application layer; it helps users express their requirement easily; it is convenient to make query optimization through in-network processing [9]. However, the disadvantage of this mechanism makes it less attractive. The effectiveness of the data collection is low. Every node needs to wait for the responses from other nodes even if those nodes may not fulfill the query conditions. Such loss in efficiency is not acceptable in wireless sensor networks which are extremely sensitive to power consumption.

Another major approach of data fusion is aggregation tree. Greedy incremental tree (GIT) is one of the most popular aggregation tree algorithms [10]. The tree is created gradually through first building the trunk of the tree and then adding different leaves. The initial tree only has a shortest route between the aggregating node and its closest source nodes. The algorithm then selects the closest node to connect to from the remaining source nodes until all the nodes are connected to the tree. The advantage of the tree algorithm is that it is well organized especially in event-driven applications. It can efficiently aggregate data while saving energy effectively. The problem with this approach is that it is difficult to fulfill all the assumptions in the real circumstance. For most sensor networks, the number of the nodes reaches magnitude of thousands and the dispatching area is easily beyond the limit of the aggregation tree.

Mobile agents are processes (e.g. executing programs) that can migrate from one machine to another machine (usually in the same system) in order to satisfy requests made by their clients [11]. Mainly, a mobile agent executes on a machine that hopefully provides the resource or service that it needs to perform its job. If the machine does not contain the needed resource/service, or if the mobile agent requires a different resource/service on another machine, the state information of the mobile agent is saved in pre-defined manner first, then the transfer to a machine containing the necessary resource/service is initiated, and the mobile agent resumes execution at the new machine. Advantages of using MAs include low network bandwidth since they only move when they need to continue execution even when disconnected from the network (typically for disconnected mode), ability to clone itself to perform parallel execution, easy implementation, deployment, and reliability.

There are many additional good reasons to use mobile agent in an wireless sensor networks environment [12]: Mobile agent reduces wireless sensor network load. Mobile agents allow users to package a conversation and dispatch it to a destination host where interactions take place locally. Mobile agents are also useful when reducing the flow of raw data in the network. When very large volumes of data are stored in remote hosts, that data should be processed in its locality rather than transferred over the network. Mobile agent overcomes network latency. Controlling sensors through a sensor network of substantial size involves significant latencies. For critical wireless sensor networks, such latencies are not acceptable. Mobile agents offer a solution, they can be dispatched from a central controller to act locally and execute the controller's directions directly. Mobile agent encapsulates protocols. When data is exchanged in a distributed system, each host owns the code that implements the protocols needed to properly code outgoing data and interpret incoming data. However, as protocols evolve to accommodate new requirements for efficiency or security, it is cumbersome if not impossible to upgrade protocol code properly. As a result, protocols often become a legacy problem. Mobile agents, on the other hand, can move to remote hosts to establish channels based on proprietary protocols. Mobile agent executes asynchronously and autonomously. Mobile motes often rely on expensive or fragile network connections. Tasks requiring a continuously open connection between a mobile device and a fixed network are probably not economically or technically feasible. To solve this problem, tasks can be embedded into mobile agents, which can then be dispatched into the network. After being dispatched, the agents become independent of the process that created them and can operate asynchronously and autonomously. The mobile device can reconnect at a later time to collect the agent. Mobile agent adapts dynamically. Mobile agents can sense their execution environment and react autonomously to changes. Multiple mobile agents have the unique ability of distributing themselves among the hosts in the network to maintain the optimal configuration for solving a particular problem. Network computing is fundamentally heterogeneous, often from both hardware and software perspectives. Because mobile agents are generally computer- and transport layer-independent (dependent on only their execution environments), they provide optimal conditions for seamless system integration. Mobile agents are robust and fault-tolerant. Mobile agents' ability to react dynamically to unfavorable situations and events makes it easier to build robust and fault tolerant distributed systems. If a host is being shut down, all agents executing on that machine are warned and given time to dispatch and continue their operation on another host in the network.

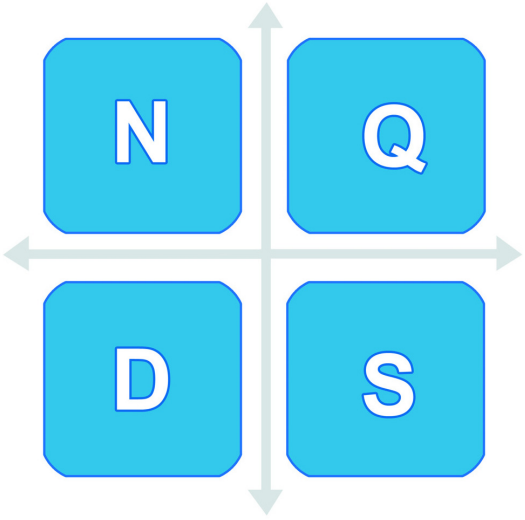


Fig. (2). The four key elements of a mobile agent.

METHODS EMPLOYED

Ordinary mobile agents will have three composing elements: code, data and state. Such configuration is very common for agents deployed in wired networks and wireless networks. However, the wireless sensor networks are far more susceptible to power consumption and band efficiency. The structure of agents should be able to adapt to the needs of complex routing and dynamic re-organization. Thus, the internal expression of the mobile agent can be designed to reflect such difference as illustrated in Fig. (**2**). In the expression above, N represent the node that the agent currently resides. The value of the variable is the coordinate of a specific node in the virtual space. This node information will guide the agent to split or merge in the treelike organized sensor network. Variable Q represents the query expression which sets the condition of the query. For example, a wireless sensor network used to monitor temperature of hotel rooms may have query expression like:

SELECT Room, MAX (Temp)
FROM Sensors
WHERE Floor = 6
GROUP BY Room
HAVING Temp > 25

D is the data collected during the traverse on the specific tree path. It is used to keep the newest results that the agent absorbed in the previous nodes. The form of the results is a two-column table which the first column contains the coordinate of a node and the second column contains the value sensor collected on that node. Only sensor data fulfilled expression Q can remain in this table. If any new sensor data is more appropriate then the current records, the table will updates itself using the new one to act as a data filter. S is the state of the mobile agent. There are five states set for the agent to reflect its life cycle: initial, search, return. In the initial state, the mobile agent starts its life in the memory and all the information sectors are set empty. In the search state, the agent record the path of traversing nodes in the form of coordinate list. In the return state, the agent evaluates the data from every node along the backtracking path.

One of the major characteristic of wireless sensor network is its dynamical structures. Power of old nodes is fading and new nodes are added into the network. It is impossible to treat the sensor networks as traditional networks. Based upon GEM (Graph Embedding) routing [13], the network first build up a virtual coordinate system and differentiate the aggregating node, leaf nodes and ordinary nodes. When the user initiates a query to the wireless sensor network system, mobile agents are created and sent out along the closest branches under aggregating node. The first task of those agents is to detect the boundary of the network. When each agent arrives at the next layer node, it first checks whether this node is a leaf node or not. If there is still a subtree, the agent will be cloned into a number of copies which equals to the number of braches under that node. Such process is repeated until the mobile agent reaches the bottom of the tree, that is, the boundary of the network. In such process, the agent also records every node it has passed into A. The purpose of doing so is to allow agents to backtrack to the aggregating node after they finish their data collection jobs. Fig. (**3**) illustrates this outgoing process.

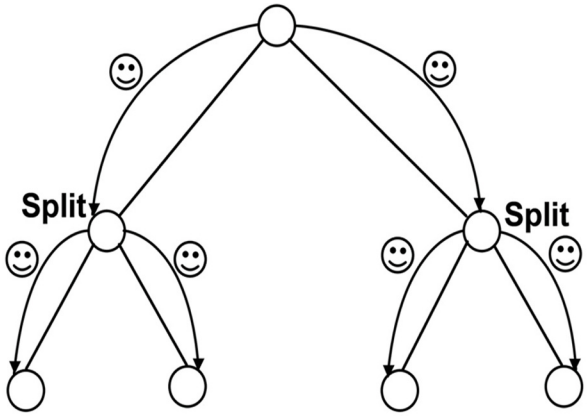


Fig. (3). Mobile agents sent to detect the boundary of the network.

Once the mobile agent reaches the boundary of the sensor network, it starts data collection through initializing a local comparison according to the Q expression embedded in it. If the sensor data fulfill the condition set, the agent move backwards with the results along its memorized backtrack path. Otherwise, the agent still moves backwards along the node list but without any results. In each node of the middle layer, the agent will check whether another agent has been waiting there. If there is an agent from other branch that has already arrived at this node, the agent that have recently arrived will send all of its D part to the earlier agent. The earlier agent will now combine its D part and the new D part. Such agent can be called as delegation agent. If an agent finds out that there is no delegation agent at all. It will become delegation agent. The delegation agent is responsible for combining data from different agents returned from different branches under a specific node. This process is illustrated in Fig. (**4**).

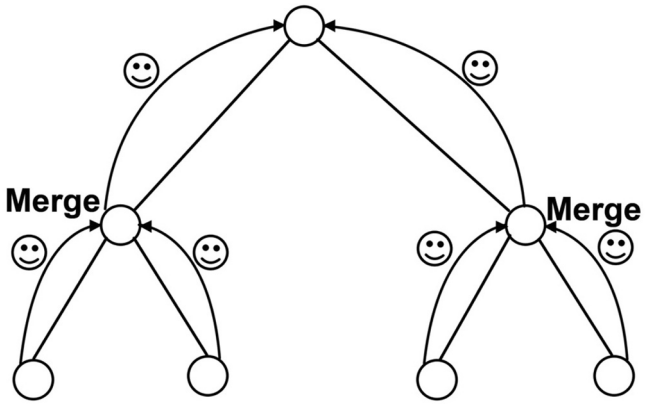


Fig. (4). Mobile agents return to the aggregating node with data collected and merged.

DATA MANAGEMENT

The major task of a wireless sensor network is to collect data from thousands of sensors. How to effectively manage

and exploit the data collected becomes one of the most critical challenges. There are many components dealing with data gathering, data correlation of static and dynamic datasets, and presenting appropriate data to end-users. Important components include: data collection; data integration; data analysis; and data presentation.

Data Collection

This involves gathering data such as: readings from wireless temperature sensors; data input by staff through their PDAs and PCs; temperature records held on servers.

Data Integration

The data management mechanism must dynamically manage and integrate static and dynamic environment data in a context and situation aware manner.

Data Analysis

The data management system generates value-added information based on the data and the meaning of this data in relation to its previous and current context. An ontology model is built using Protégé. Expert system rules using Jess [14] can be applied to manage the context variables. The aim of this component is to intelligently analyze the large amount of application data. This provides the basis for improved control over building air conditioner control.

Data Presentation

Data is filtered and, depending on the current context aware variables, relevant temperature data is sent to the end user.

Agent platforms have been applied in a number of WSN environments [15] and [16]. The integration of software agents is encouraged by the prediction expressed in [17] "that mobile agents capable of discovering, extracting, interpreting and validating context will make significant contribution to increasing efficiency, flexibility and feasibility of pervasive computing systems".

Most of the components within the data management system are implemented with the more sophisticated JADE agent platform. It is clear that Agilla is appropriate for the important task of data collection from the wireless temperature sensor nodes. The basic function of the data collection agent is to retrieve sensor readings from the wireless temperature sensor nodes. One may configure the agent to take readings at regular intervals. However during temperature monitoring the rate at which these readings are taken may need to be altered. For example, the temperature's condition may change or medical staff may make a reading request. The agent paradigm is ideally suited to the reactive/proactive nature of temperature monitoring. Therefore extending the agent architecture to the actual temperature sensor nodes is essential for a comprehensive solution.

THE XML REPRESENTATION OF THE DATA

XML Schema is used to depict meta-data of mobile agent and the topology of the network [18]. This is to define a descriptive standard so that agents can communicate with each other in a common language. They may compare, combine and exchange the information about sensor collected data, network topology and transportation status. The schema defines three categories of information that may appear in a loosely connected sensor network.

The Path specific schema defines traversed nodes, node-node power consumption, initial node, end node. The traversed nodes are a list which records every node that the agent has passed through during its journey toward the end node. They are also the guidance that leads the agent move back to its initial node, that is, the aggregating node. The node-node power consumption records the power cost between two adjacent nodes. This is an important criterion that helps an agent make decision when choosing path to return. The initial node and end node represent the aggregating point of sensor network and the boundary of the sensor network respectively.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
targetNamespace=http://www.example.com/IPO
xmlns:ipo="http://www.example.com/IPO"
xmlns:xsd="http://www.w3.org/2000/10/XMLSchema">
<xsd:element name="traversed_nodes" type="xsd:string"/>
<xsd:element name="power_consumption"
type="xsd:string"/>
<xsd:element name="initial_node" type="xsd:integer"/>
<xsd:element name="end_node" type="xsd:integer"/>
</xsd:schema>
```

The sensor specific schema depicts some critical information that composes of the mobile agent itself: sensor identity, sensor type, and sensed value. The sensor identity uniquely defines the sensor node by which the agent passes. The sensor type defines the varieties of types of the sensors. This is useful when a sensor network is made from many different types of sensors. For example, a water monitoring sensor network may contain both temperature sensors and PH sensors [19]. The sensed value records the value collected before or at the time when the agent stays in that sensor node. This is also the target of data fusion when the agent moves back toward its aggregating point.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
targetNamespace=http://www.example.com/IPO
xmlns:ipo="http://www.example.com/IPO"
xmlns:xsd="http://www.w3.org/2000/10/XMLSchema">
<xsd:element name="sensor_id" type="xsd:integer"/>
<xsd:element name="sensor_type" type="xsd:string"/>
<xsd:element name="sensed_value" type="xsd:double"/>
</xsd:schema>
```

To describe a mobile agent that works in a sensor network, it is also necessary to create some agent descriptive schema including agent identity, agent task, and life cycle. The agent identity uniquely marks the agent and allows agents communicating with each other in a named manner. The agent task is the condition of collecting data which is pre-assigned in the initial node; it is used as a filter to decide which data fulfills the requirement of the aggregating point. The concrete presentation of the task is in the form of SQL like language which is embedded into the XML description.

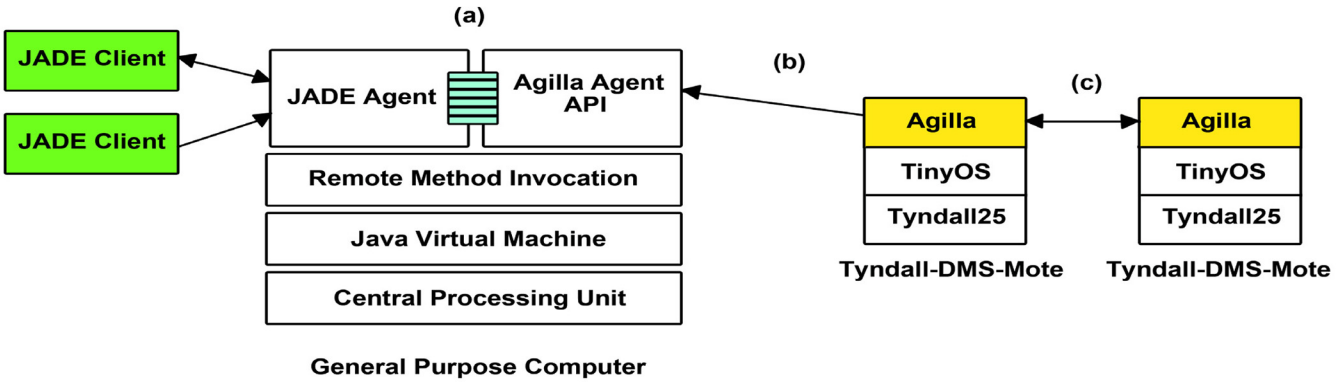


Fig. (5). The integration of Jade agent and Agilla agent [22].

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
targetNamespace=http://www.example.com/IPO
xmlns:ipo="http://www.example.com/IPO"
xmlns:xsd="http://www.w3.org/2000/10/XMLSchema">
<xsd:element name="agent_identity" type="xsd:integer"/>
<xsd:element name="agent_task" type="xsd:string"/>
<xsd:element name="life_cycle" type="xsd:integer"/>
</xsd:schema>
```

NEGOTIATION BASED DATA FUSION

When two or more agents meet in some branch node, a negotiation is organized to filter out redundant data, recalculate the optimum path back toward the initial point, and choose representative agent to delegate the other agents. The following list describes the mechanism that is used in the negotiation among agents.

First — An agent arrives at a node called N, the agent first check the registry R to see whether there are another agent currently reside in the node.

Second — if there is an agent who does reside in the registry of the node, the agent sends its own agent identity as a request to the target agent. The target agent responds to the initiate agent with their identities as reply. Thus a communication channel is built between these two agents. The communication language is based upon ACL [20] in which the form of contents make use of the XML described in the above section.

Third — The agent sends a new request which require the target agent to send a copy of its agent internal data, path data, and sensor data. Once received the copy arrives, the agent compares the task of the target agent with its own task. If they have the same task, the agent continues to compare their sensed data according to the condition in the task. Such comparison could simply base upon a value comparison or base upon a complex logic composition. It all depends on the specific application domain.

Forth — if one of the agent exhibits one or more sensed data which better fit for the task condition, this agent is chosen as the representative of these two agents. The representative agent will keep its qualified data and rearrange its path back toward the aggregating point according to another agent's path information that is acquired during the negotiation. The agent that fails to be selected as the representative will cease to live and let the representative agent fully delegates it.

Fifth — The rearranging of path is a process of optimizing the return path toward the aggregating point. The node-node power consumption embedded in the XML file of the two agents records the power cost map during their journey. The power cost information is very important information because the wireless sensor network is deeply sensitive to the power. The representative agent will adopt the less power cost route from two candidate route, its own path and the target's path.

Fig. (**6**) illustrates the optimization process. The representative agent will reject its original route A to aggregating point and choose Route which is originally belong to the cancelled agent as the new path back to aggregating point because the power consumption of Route A is less than the power consumption of Route B. Thus B now is the new path of the representative agent.

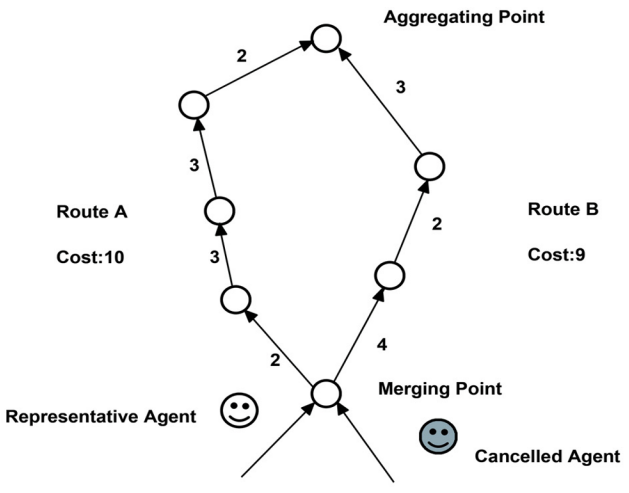


Fig. (6). The Optimization of Return Path.

INTEGRATION OF AGENT DOMAINS

The approach designed for the target application makes use of both mobile agent paradigm and multi-agent paradigm. The reason of choosing both paradigms is that the target application domain encompasses fix nodes such as PCs and Server as well as mobile nodes such as Mica2 wireless sensors. There are big difference on their hardware resources, specifications and usages. Generally speaking, the mobile nodes undertake some simpler tasks with less resource while the fixed nodes undertake some complex tasks with more computing capabilities. In our approach, Jade is selected as the implementation tools for fixed nodes; Agilla is selected as the implementation tool for mobile nodes. The following part of this section will briefly discuss the advantages and disadvantages of those two tools and the integration of those two tools.

Java Agent DEvelopment Framework (JADE) [21] is Java based agent platform developed by Tele-com Italia Lab. It is an open source project under GNU lesser GPL. It is fully FIPA compliant and on top of that it supports all FIPA message encodings (XML, Lisp, and binary). There is a direct sup-port for SL, RDF, and XML content languages. It supports multiple containers that can be running on different machines forming one agent platform. FIPA compliant White and Yellow pages services are available. There is a support for an interconnected set of these services defined by the user. A subscription for these services is available too. There is a great collection of add-ons and 3rd party software available since there is a huge community of developers that are using JADE. The architecture of Jade platform is illustrated in Fig. (**7**).

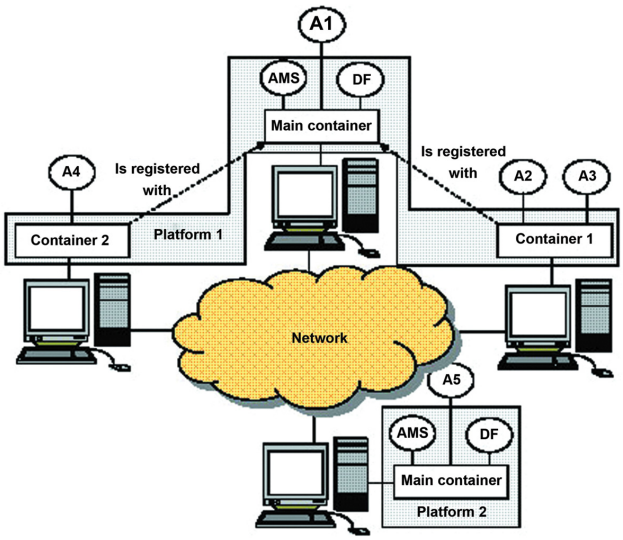


Fig. (7). The architecture of JADE platform.

For the MAS research community, it is possible to develop better tools and platforms to support the agent development through three routes. The first route is to update the FIPA specification with continuously absorbing new findings and new mechanisms, to incorporate with the existing widely adopted specifications such as XML and UML; the second route is to build more powerful application platform through strengthening the administration, monitoring, debugging and logging functionality, to exploit the successful development tools like eclipse, to incorporate Web Service and agent oriented software engineering; the third route is to maintain and enlarge the open source communities and let more researchers take part in and contribute to the development of platforms.

Agilla [22] is a middleware for wireless sensor networks (WSNs) that provides a mobile-agent style of programming. Agilla applications consist of mobile agents that can proactively migrate their code and state across the network. Mobile agents offer more flexibility by allowing applications to control the way they spread. They can position themselves in the optimal locations for performing application-specific tasks. They can save energy by bringing computation to the data rather than requiring that the data be sent over unreliable wireless links to the location containing the computation. They can increase the utility of a WSN by constraining themselves to the specific locales that are relevant to their application's requirements (in contrast to spreading throughout an entire network), and sharing the resources of a single node, i.e., multiple mobile agents can reside on each WSN node. Other systems like Deluge and Maté allow in-network reprogramming. Agilla, however, allows programs to control where they go and to maintain both their code and state across migrations.

Agilla provides a stack-based architecture for each agent, as shown in in Fig. (**10**). This reduces overhead by allowing most instructions to be one byte. To allow inter-agent communication, Agilla maintains a tuple space on each node that is shared by all agents residing on the node, and is accessible remotely. By interacting through a tuple space, each agent remains autonomous, decoupled both spatially and temporally. To minimize the impact of message loss, agents are divided into tiny packets (< 41 bytes), are migrated a single hop at a time, and utilize timeouts and retransmits. Since this hop-by-hop process introduces a significant amount of store-and- forward delay, it is only used while migrating or cloning agents, not for remote tuple space operations. Remote tuple space operations are non-blocking, preventing an agent from deadlocking due to message loss. In recognition of the importance of spatial data within sensor networks, Agilla addresses nodes by their location. All remote operations take

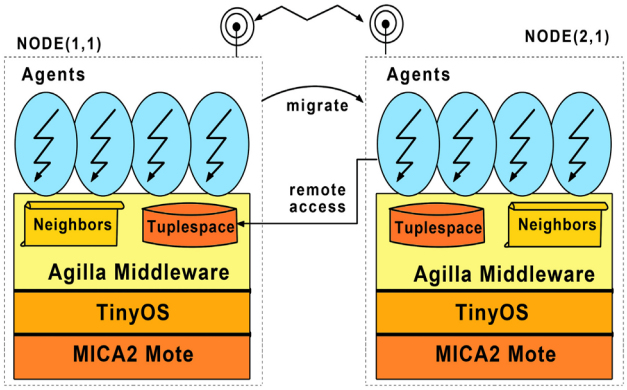


Fig. (8). The working process of Agilla middleware [22].

a location as a parameter. For example, instead of cloning to a node with id=1, an agent would clone to a node at location (1,1). By tailoring Agilla's primitives to sensor networks, Agilla provides a foundation for rapidly developing applications for WSNs with unprecedented levels of flexibility. The working process and structure of Agilla is illustrated in Figs. (**8** and **9**).

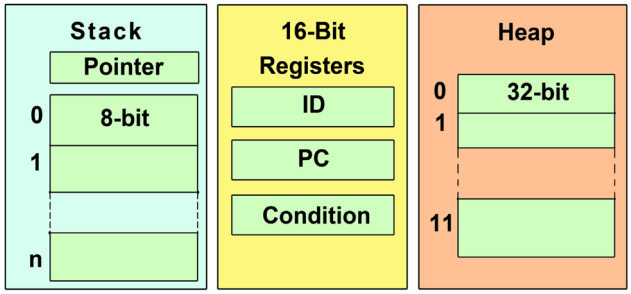


Fig. (9). the structure of Agilla (Herbert WIIAT 2006) [22].

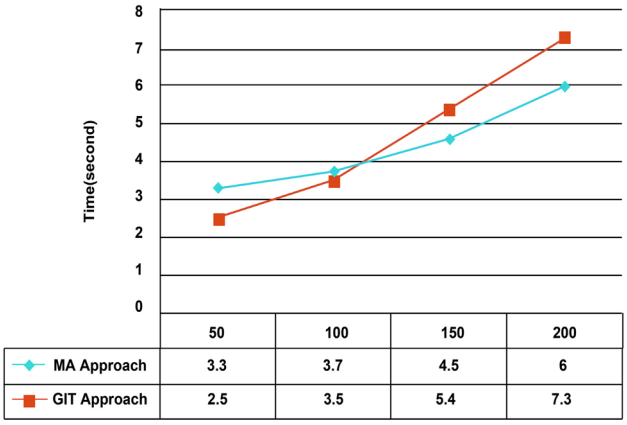


Fig. (10). Topology Control Efficiency under Different Number of Nodes.

Since new agents can be injected into a pre-existing network, the network can be re-tasked. Since each agent executes autonomously and multiple agents can simultaneously run on a node, multiple applications can co-exist. Since mobile agents can move and clone, they can quickly morph an application's installation base to handle unexpected changes in an environment. There are many other inherent advantages of using mobile agents, especially in a wireless sensor network.

However, there are also many challenges, the foremost being the lack of resources and an unreliable network. Agilla is the first mobile agent middleware for WSNs that is implemented entirely in TinyOS. It has been tested on Mica2, MicaZ, and Tmote Sky motes. It hides complexities associated with developing WSN applications, and provides mechanisms that overcome the challenges associated with limited resources and unreliable network communication. It demonstrates the feasibility of using mobile agents within WSNs and, furthermore, takes the first steps towards identifying a minimal set of primitives that should be provided for facilitating highly flexible WSN applications.

The concept of integrating a pure agent platform with Agilla was first introduced in [23], with work also reported in [24]. Implementation of a JADE-Agilla integration must fulfil the following minimum requirements: communication *via* an interface; agent injection from resource rich JADE to the restricted Agilla environment; migration of agents in the sensor network; retrieval of sensor readings by Agilla agents and return of results to the main JADE environment. JADE to Agilla interaction is achieved *via* method calls defined in a generic interface (cf. Fig. **2**). This provides a straightforward way of injecting Agilla agents. While arbitrary Agilla code can be injected into the wireless sensor network it is useful to have a number of Agilla agents to perform standard tasks such as retrieving sensor readings at regular intervals, or retrieving sensor data in response to an asynchronous request.

Combining agent injection with a library of useful Agilla agents extends the agent architecture to the wireless sensor nodes in a seamless manner. A library of the most common functions has been created. Examples include functions to retrieve temperature data periodically and to take an immediate blood pressure reading.

RESULTS

The researcher builds an experimental platform in the laboratory environment based upon Mica2 [25] series nodes and Tiny OS [26] operating system. The MICA2 Mote is a third generation mote module used for enabling low-power, wireless, sensor networks. The MICA2 Mote features several new improvements over the original MICA Mote. The MPR400 is based on the Atmel ATmega128L. The ATmega128L is a low-power microcontroller which runs MoteWorks from its internal flash memory. A single processor board (MPR400) can be configured to run your sensor application/processing and the network/radio communications stack simultaneously. The MICA2 51-pin expansion connector supports Analog Inputs, Digital I/O, I2C, SPI and UART interfaces. These interfaces make it easy to connect to a wide variety of external peripherals. Fig. (**11**) illustrates the MPR 400 block diagram.

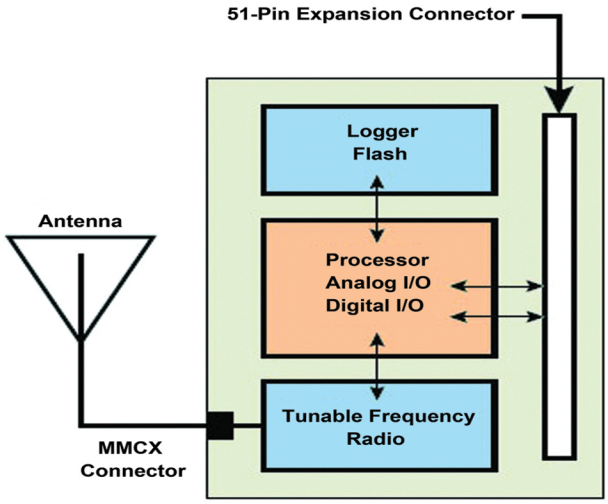


Fig. (11). MPR400 Block Diagram [27].

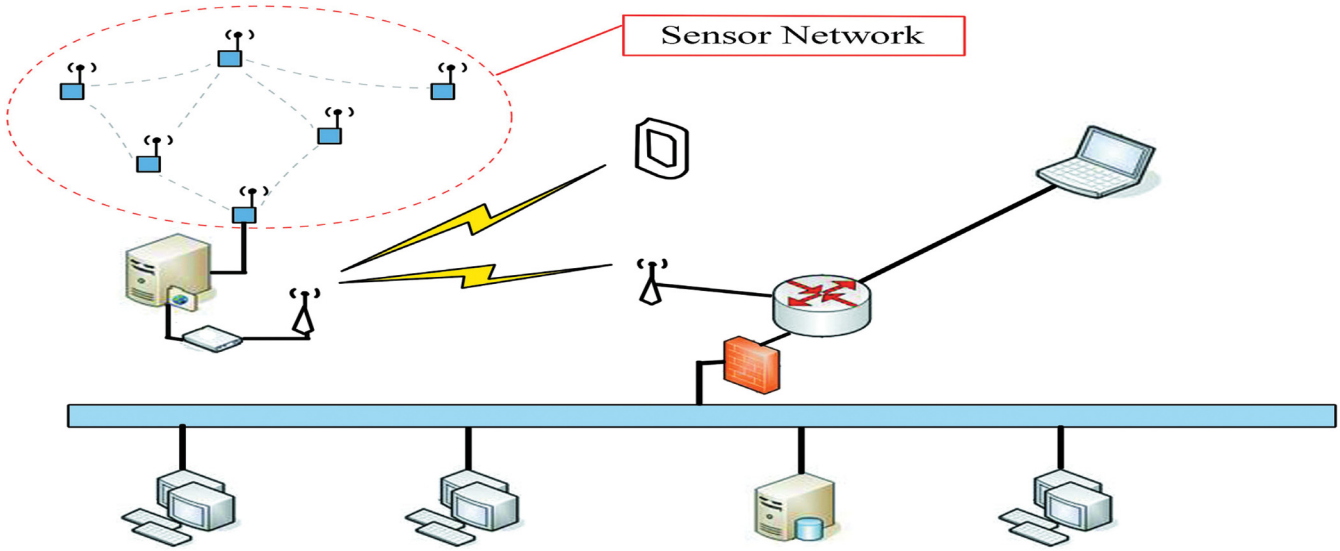
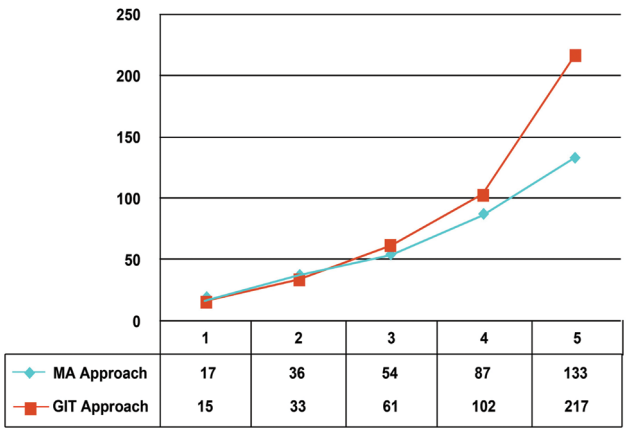


Fig. (12). The experimental environment design.

The application scenario is a small scale temperature monitoring system which encompasses the area of 1000 square meters. The wireless sensors are dispatched randomly at any location in the confined area. The mobile agent is designed with the nesC [28] programming language which supplies good component-based and event-driven mechanisms. The design of experiment environment is illustrated in Fig. (**12**).

Fig. (**5**) is the result of comparing the mobile agent based approach with the popular GIT approach [29]. In this comparison, the time starts from a message sent out or an agent initialized, ends at the message loops back to the aggregating point or an representative agent arrives at the aggregating point. From this result, it is found that the mobile agent approach is less efficient than the GIT approach when the number of nodes in the wireless sensor network is less than 100. However, when the number of nodes exceeds 100, the advantage of mobile agent based approach becomes to emerge. For example, when number of nodes reach 200, the time used in MA approach is 6 second, while the GIT approach use 7.3 second. This phenomenon is illustrated in Fig. (**10**). If the experiment last longer, the difference between these two approaches will be more significant.

Because the path optimization is designed for the MA approach, the researcher also carries out an experiment to evaluate the effectiveness of such mechanism. Fig. (**13**) illustrates the result of the comparison between two approaches under the power consumption criterion. The duration of the experiment had lasted for 5 months. The research sets three of the sensor nodes as the observing points. Every month, their power costs are recorded and averaged to produce a record. This record is then summed up to the corresponding record in the following figure. It is found that there is only minor difference on the power consumption for both two approaches in the first three months. In the long run, the MA approach exhibits better power saving ability especially in the fifth month. The power consumption of MA approach is only 61.3% of the GIT Approach in a 5 month's duration.



	1	2	3	4	5
MA Approach	17	36	54	87	133
GIT Approach	15	33	61	102	217

Fig. (13). Result of the comparison between two approaches under the power consumption criterion.

Through the above two comparison, a summary for the new mobile based data fusion approach can be listed as follow:

The efficiency of mobile based data fusion approach is better than the other approaches especially when the node of network exceeds a threshold of 100. This phenomenon implies that the mobile based data fusion approach can achieve higher data fusion rate than the other data fusion approaches.

The optimized path rearrangement introduced in the mobile based data fusion approach can significantly reduce the power consumption in the long run. Such path optimization also contributes to the efficiency of the network because the agent can decide the shortest and power-cost least way dynamically according to the other agents' experiences.

DISCUSSION

The XML representation of the agent internal framework, the track history and the sensor specific data do give the proposed approach a standard and strong expressive way to describe the verities of information related. However, on the other side, the processes of parsing a XML file do introduce another layer of burden that may decrease some levels of efficiency. This is also the reason that the MA approach is less efficient when the number nodes are less than 100. To alleviate the deficient of the XML parsing, the researcher carefully chooses the information elements that may appear in the design. The researcher also tries to avoid some unnecessary redundant description in a normal XML file. Additionally, an embedded version XML parser is implemented in nesC language which excludes many unused features like ordinary Java or C++ XML related API [30]. This parser can be imported into the wireless sensor chip and parses the custom defined XML file efficiently.

Another issue deserves to be discussed is the problem of platform independence. Currently many wireless sensor networks are based upon private technology. For example, in the researcher's experiment, nesC and Tiny OS is used to support the design and implementation. However, a wireless sensor network could be a network of heterogeneous sensors, that is, the sensors may come from different sources and may contain very different inner structures or physical implementations. It is not practical to base the sensor networks upon the technology that only functions in one specific platform. In this article, the researcher tries to use the standard data expressive way to build common understanding fundamentals for different types of wireless sensor network. It is also necessary to transform such commonality to the development platform, that is, to build a cross platform solution for varieties of wireless sensor networks. A specially designed Java with appropriate API and IDE support could be a future choice [31].

It is possible to extend the mobile agent based wireless sensor network to other scenarios such as fire extinguishing, environment monitoring. The key point of extension is how to build a common vocabulary appropriate for different context that is, building a framework which can support different application domains. In such framework, the same sensor can play multiple sensor roles in case the mobile agent traverse those nodes can interprets the semantics of the received data. Ontology and semantic web development do show a future of intelligent program which can understand the semantic behind specific target object. Ontology defines a common vocabulary for researchers who need to share information in a domain. It includes machine-interpretable definitions of basic concepts in the domain and relations among them.

The reasons of using ontology in a data fusion problem solving are listed as followed [32]:

- To share common understanding of the structure of information among people or software agents
- To enable reuse of domain knowledge
- To make domain assumptions explicit
- To separate domain knowledge from the operational knowledge
- To analyze domain knowledge

Researchers do not need to waste time to build their own tools of ontology implementation. Protégé is a free, open-source platform that provides a growing user community with a suite of tools to construct domain models and knowledge-based applications with ontologies. At its core, Protégé implements a rich set of knowledge-modeling structures and actions that support the creation, visualization, and manipulation of ontologies in various representation formats. Protégé can be customized to provide domain-friendly support for creating knowledge models and entering data. Further, Protégé can be extended by way of a plug-in architecture and a Java-based Application Programming Interface (API) for building knowledge-based tools and applications.

The Protégé's approach makes use of an iterative approach to ontology development [33]: The approach start with a rough first pass at the ontology. Then it revises and refines the evolving ontology and fill in the details. Along the way, the modeling decisions that a designer needs to make are discussed, as well as the pros, cons, and implications of different solutions.

Determine the domain and scope of the ontology: The domain that the target application will cover is an intelligent building that is facilitated with thousands of wireless temperature sensors. The domain should also include the server which run the air conditioner controller and the PCs which run the monitor client. If the mobile agent based system is applied in a different domain whose ontologies have been already defined, the existing ontologies can be reused so that researchers can fully appreciate the heritage of the legacy system and the work has been done before. Once the domain has been established, terms inside the domain can be enumerated. For each of the term, there would be some properties which describe attributes of the term. For different terms, the researcher can analyze what are the relationships among them, and how one term may influence the other term. The remaining jobs are using the functionalities provided by Protégé, that is, define the classes and the class hierarchy for the terms in the target domain. Naturally, the properties of each class should be specified as well. In Protégé, such properties are also called slots. Slots can have different facets describing the value type, allowed values, the number of the values (cardinality), and other features of the values the slot can take. The last step is creating individual instances of classes in the hierarchy. Defining an individual instance of a class requires choosing a class, creating an individual instance of that class, and filling in the slot values.

CONCLUSION AND FUTURE WORK

Introducing mobile agent into the field of data fusion for wireless sensor network is valuable because the communication ability of mobile agent can fully facilitate the information sharing among nodes in a sensor network. The article proposes a new approach which use mobile agent as basic unit to collect data and process data. The method also employs XML representation to express data and agent framework. The advantages of the approach can be summarized here:

1) The efficiency of mobile based data fusion approach is better than the other approaches.
2) The XML representation of the agent internal framework, the track history and the sensor specific data do

give the proposed approach a standard and strong expressive way to describe the verities of information related.

3) The optimized path rearrangement introduced in the mobile based data fusion approach can significantly reduce the power consumption in the long run.

In the future, it is possible to improve the approach from three points:

1) Transform the mobile agent from necC implementation to a Java implementation, thus improve the platform independence of the method.

2) Re-construct the XML parser inside the agent framework so that the parsing and processing of XML could be quicker and more efficient.

3) Introduce synchronous mechanism into the mobile agent, so that the agent can control its steps traversing back to the aggregating point. The agents can have more time to communicate and share their knowledge about the network.

REFERENCES

[1] F. Akyildiz, W. L. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor network," *IEEE Commun. Mag.*, vol. 40, pp. 102-114, 2002.

[2] W. R. Heizelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proc. 33rd Hawaii Intl. Conf. Syst. Sci., (HICSS'00)*, 2000, pp. 1-10.

[3] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks," in *Proc. 6th Annu. ACM/IEEE Intl. Conf. Mobile Comput. Network., (MobiCOM'00)*, Boston, MA, August 2000.

[4] A. Manjeshwar, and D. P. Agrawal, "TEEN: A routing protocol for enhanced efficiency in wireless sensor networks," in *Proc. 15th Intl. Parallel Distributed Process Symp.*, (IPDPS'01), San Francisco, CA, April 2001.

[5] D. Estrin, R. Govindan, J. S. Heidemann, and S. Kumar, "Next century challenges: Scalable coordinate in sensor network," in *Proc. 5th ACM/IEEE Int. Conf. Mobile Comput. Network.*, 1999, pp. 263-270.

[6] *WINS: Wireless integrated network sensors*. Los Angeles: University of California. http://www.janet.ucla.edu/WINS/biblio.htm [Accessed Feb.18, 2009].

[7] T. He, S. Krishnamurthy, J. A. Stankovic, T. Abdelzaher, L. Luo, R. Stoleru, T. Yan, L. Gu, J. Hui, and B. Krogh, "An energy-efficient surveillance system using wireless sensor networks," in *Proc. MobiSys., 2004*, June 2004.

[8] P. Rentala, R. Musunuri, S. Gandaham, and U. Saxena, "*Survey on sensor networks*," University of Texas, Dallas, Tech. Rep. UTDCS-33-02, 2002.

[9] Y. Yao, and J. E. Gehrke, "The cougar approach to in-network query processing in sensor network," *ACM Sigmod. Record.*, vol. 31, pp. 9-18, 2002.

[10] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "*TAG: A tiny aggregation service for ad-hoc sensor network,*" in *Proc. Oper. Sys. Des. Implement.*,(OSDI), USENIX, December 2002.

[11] B. Krishnamachari, D. Estrin, and S, Wicker, "Impact of data aggregation in wireless sensor networks," in *Proc. IEEE INFOCOM*, 2002, pp. 575-578.

[12] D. Massaguer, "*Multi mobile agent deployment in wireless sensor networks*," Master's thesis, University of California, Irvine, 2005.

[13] J. Newsome, and D. Song, "GEM: Graph embedding for routing and data-centric storage in sensor networks without geographic information," presented at the *1st ACM Conf. Embed. Network. Sensor Sys. (SenSys'03)*, Redwood, CA, November 2003.

[14] JESS: the Java Expert System Shell, 2008. [Online]. Available: http://herzberg.ca.sandia.gov/jess/ [Accessed Dec. 12, 2008].

[15] C. L. Fok, G. C. Roman, and C. Lu, "Rapid development and flexible deployment of adaptive wireless sensor network applications," in *Proc. 24th Intl. Conf. Distrib. Comput. Sys. (ICDCS'05)*, 2005, pp. 653-662.

[16] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein, "Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet," presented at the *Int. Conf. Archit. Support Program. Languages Operat. Sys.*, San Jose, CA, USA, Oct. 2002.

[17] N. Xu, S. Rangwala, K. K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin, "A wireless sensor network for structural monitoring. SenSys '04," in *Proc. 2nd Intl. Conf. Embed. Network. sensor Sys.*, 2004, pp. 13-24.

[18] http://www.w3c.org.2008

[19] http://www.atmel.com/atmel/acrobat/1065s.pdf 2008.

[20] J. M. Kahn, R. H. Katz, and K. S. J. Pister, *Next century challenges: Mobile Networking for "Smart Dust"*, Department of Electrical Engineering and Computer Science, University of California, Berkeley, 2006.

[21] F. Bellifemine, A. Poggi, and G. Rimassa, "JADE – FIPA Compliant Agent Framework," in *Proc. PAAM*, 1999, pp. 97-108.

[22] J. Herbert, J. O'Donoghue, G. Ling, K. Fei, and C. Fok, "Mobile Agent Architecture Integration for a Wireless Sensor Medical Application" 2006 IEEE/WIC/ACM *Int. Conf. Web Intelligence Intell. Agent Technol. - Workshops*, 2006, pp. 235-238.

[23] D. Moore, J. Leonard, D. Rus, and S. Teller, "Robust distributed network localization with noisy range measurements," *Second ACM Conf. Embed. Network. Sensor Syst. (Sensys 04)*, November 2004.

[24] P. Kang, C. Borcea, G. Xu, A. Saxena, U. Kremer, and L. Iftode, "Smart messages: A distributed computing platform for networks of embedded systems," Special Issue on Mobile and Pervasive Computing, *Comput. J.*, vol. 47, Oxford University Press, 2004.

[25] B. Paul, "Real-Time Communication and Coordination in Wireless Embedded Sensor Networks," SYSC *5710-Operating System Methods for Real-Time Applications*, 2003.

[26] R. Gupta, and S. R. Das, "Tracking moving targets in a smart sensor network," The VTC Fall 2003 Symposium, October 2003.

[27] Mica Mote, *MICA: Wireless Measurement System*, [Online]. Available: http://www.xbow. com/Products/Product_pdf_files/Wireless_pdf/MICA.pdf [Accessed Nov. 22, 2008].

[28] A. Boukerche, X. Cheng, and J. Linus, "Energy-aware data-centric routing in microsensor networks," in *Proc 6th ACM Intl. Workshop Modeling Anal. Simul. Wireless Mobile Sys. (MSWiM'03)*, San Diego, Sep 19, 2003, 42-49, CA.

[29] D. Gay, P. Levis, D. Culler, and E. Brewer, *nesC 1.1 Language Reference Manual*, 2003.

[30] Tiny OS, [Online]. Available: http://www.tinyOS.net/ [Accessed Sep. 20, 2008].

[31] Maxim. [Online]. Available:http://www.maximic.com.cn/appnotes 10.cfm/ac_pk/38/ln/cn [Accessed Nov. 5, 2008]

[32] L. Tong, Q. Zhao, S. Adireddy, "Sensor Networks with Mobile Agents", *In Proc. 2003 Military Commun. Intl. Symp.*, Boston, MA, 2003.

[33] N. F. Noy, M. Sintek, S. Decker, S. Crubezy, M. Fergerson, and R. W. Musen, "Creating Semantic Web Contents with Protege-2000," *IEEE Intl. Sys. Appl.*, vol. 16, issue 2, pp. 60-71, 2001.

Received: February 15, 2009 Revised: May 25, 2009 Accepted: June 03, 2009