A Data Aggregation System using Mobile Agents on Integrated Sensor Networks

Yuto Hamaguchi*, Tomoki Yoshihisa*, Yoshimasa Ishi*,Yuuichi Teranishi†, Takahiro Hara*, and Shojiro Nishio*
*Department of Multimedia Engineering,Graduate School of Information Science and Technology, Osaka University
†National Institute of Information and Communications Technology, Japan
Email: [hamaguchi.yuto,yoshihisa,ishi.yoshimasa]@ist.osaka-u.ac.jp, teranisi@nict.go.jp, [hara,nishio]@ist.osaka-u.ac.jp

Abstract—**Due to the recent development of sensor networks, integrated sensor networks, in which some sensor networks are managed systematically, have attracted considerable attention. To implement various applications, such as weather forecasting and environmental observation, with integrated sensor networks, users (clients) usually aggregate and collect the sensor data obtained from all the sensor networks. After this, the clients execute some operations, such as averaging and analysis, on the aggregated sensor data. However, such processes cause heavy network traffic in aggregating data from all the sensor networks. Hence, in this paper, we propose a data aggregation system using mobile agents. Mobile agents are programs that migrate among sensor networks. Since the mobile agents migrate while executing operations on the sensor data, the clients do not need to aggregate the sensor data by collecting it from all the sensor networks, and the network traffic can be reduced.**

***Keywords*-wireless sensor networks; mobile agents; peer-to-peer networks.**

I. Introduction

Due to the recent development of sensing technologies, sensor networks, in which sensors construct information networks and communicate with each other, have attracted considerable attention. A sensor network typically has a sink node, which collects the sensor data generated from the sensors connected to the sensor network. However, sensor networks have limits regarding the geographical area they can cover, since the sensors must function within communication ranges of other sensors. Therefore, we need to integrate local sensor networks in order to construct broad sensor networks that can cover a wide area. We call the constructed sensor networks as *integrated sensor networks*. With integrated sensor networks, users can collect sensor data obtained from various sensor networks. In the following examples, there are three sensor networks deployed respectively at Osaka, Kyoto, and Nara, which are notable Japanese cities in the Kansai area, located near one another.

- For weather forecasting, a user calculates the average temperature of these cities. To calculate the average value, the client aggregates the temperature sensor data obtained from all the sensor networks.
- For environmental observation, a user finds the hottest city among the three cities. To find the city that has the maximum temperature value, the client calculates the temperature sensor data obtained from all the sensor networks.
- To find broken temperature sensors, a user finds the sensors generating abnormal values, by analyzing the sensor data generated from all the sensor networks.

To aggregate sensor data generated from integrated sensor networks, several data aggregation systems have been proposed ([1], [2]). In these systems, users (clients) must aggregate the sensor data obtained from all sensor networks for their clients. After this, the clients execute some operations, such as averaging and analysis, on the aggregated sensor data. However, such processes cause heavy network traffic in aggregating data from all the sensor networks. To solve this problem, data aggregation operations should be performed only on necessary data at each sensor network. For example, to determine the average temperature for weather forecasting, the clients do not need to collect all the sensor data, but only the average temperature and the number of sensors for each sensor network.

Hence, in this paper, we propose a data aggregation system using mobile agents on integrated sensor networks. Mobile agents are computer programs that migrate among sensor networks. In our proposed system, clients generate mobile agents. The mobile agents execute the user-written programs migrating among sensor networks, and finally return to the clients. Since the mobile agents migrate while executing operations on the sensor data, the clients do not need to collect and aggregate the sensor data generated from all the sensor networks, and thus network traffic can be reduced.

The rest of this paper is organized as follows. We introduce related work in Section II. In Section III, we discuss the requirements for integrated sensor networks, and explain the system designed to satisfy these requirements. We describe the implementation of our proposed system in Section IV, and discuss its merits and demerits in Section V. Finally, in Section VI we conclude the paper.

II. Related Work

A number of sensor data aggregation systems for integrated sensor networks have been developed. LiveE! collects the sensor data observed by digital instrument shelters ([3]). The shelters are managed by 11 decentralized servers, and have temperature, humidity, pressure, wind-direction, wind-speed, and rainfall sensors. Since the servers are decentralized, LiveE! can relieve the servers' load. However, users must access all the servers to collect all the sensor data.

X-Sensor 1.0 is a sensor network testbed which our research group has developed ([4], [5]). X-Sensor 1.0 has several sensor networks that are deployed in major Japanese universities. By registering a sensor network with the X-Sensor 1.0 testbed server, the sensor network can be managed by the centralized server. Users can collect the sensor data from the X-Sensor 1.0 system via the centralized server. Compared with X-Sensor 2.0, it is difficult to aggregate sensor data from multiple sensor networks with the X-Sensor 1.0 since we have to submit aggregation queries to sink nodes for each sensor network.

IrisNet [2] stores the sensor data in the distributed sensor databases. In IrisNet, SAs (Sensing Agents) collect the sensor data from several sensors. The SAs aggregate the collected sensor data for nearby OAs (Organizing Agents). The OAs select the most relevant database in which to store the aggregated data, from the viewpoint of overall system performance, and transfer their data to this database.

Environmental Monitoring 2.0 is a data-sharing and visualization system using Sensormap [6] and GSN (Global Sensor Network) [7]. Sensormap provides data-sharing services, and GSN provides data-management services. In contrast to the three systems above, Environmental Monitoring 2.0 focuses on the visualization system. It can show the sensor type, sensor data, and so on, visually.

However, in these systems users must collect the sensor data for their clients; and after this, the clients execute operations on the collected sensor data. In our proposed system, the clients do not need to collect all the sensor data, since mobile agents migrate among the sensor networks while executing operations on the sensor data.

Some systems designed to manage mobile agents, such as AgentTeamwork [8], PIAX [9], and MADSN [10], have been proposed. AgentTeamwork uses mobile agents for grid computing. The mobile agents migrate to the computers that have the necessary data, and execute the required tasks there. PIAX is a P2P-based mobile-agent system. However, AgentTeamwork and PIAX do not focus on sensor data aggregation. MADSN, on the other hand, does not focus on integrated sensor networks. A mobile-agent system customized for integrated sensor networks offers users easy and efficient aggregation of sensor data generated from such networks.

III. Requirements and Design

As noted in Section I, the network traffic required to aggregate the sensor data from conventional integrated sensor networks is large, since the clients must often collect all the sensor data from all the sensor networks. By executing operations on the sensor data and aggregating only the necessary data, the network traffic can be reduced. Mobile-agent systems are suitable for executing operations on respective sensor networks, since they can migrate among the sensor networks, executing operations. Therefore, we use mobile agents.

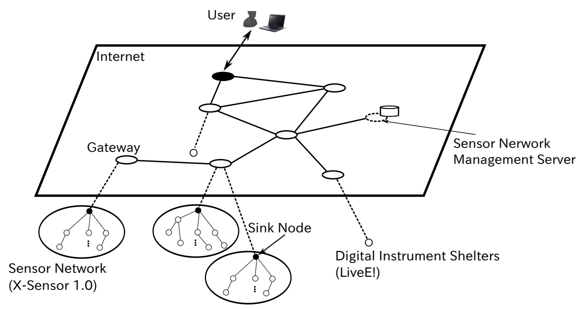

Figure 1 An Integrated Sensor Network

A. Requirements

In this subsection, we describe the requirements for data aggregation systems using mobile agents.

1) Management of Mobile Agents: To enable mobile agents to aggregate the sensor data, we need to manage the mobile agents. That is, the system generates the mobile agents and controls them according to users' operations. For example, users select the sensor networks which they want to aggregate the sensor data, and write a data aggregation program for the mobile agents. After this, the clients generate the mobile agents, and the mobile agents begin to migrate. The written program is executed in each sensor network when the mobile agents migrate to it. In addition, users can control the mobile agents by sending operations such as 'move', 'stop', and 'destroy', and debug the program in the course of the mobile agents' execution of their tasks. Finally, the mobile agents return to the client. The system must manage the mobile agents in order to accomplish such operations.

2) Visual Interface: To facilitate control of the mobile agents, it is useful to determine where they are and what they are doing. With visual access to their locations, users can intuitively determine where the mobile agents are. In addition, visual interfaces facilitate users' selections of mobile agents. Users select the mobile agents using a visual interface when they wish to confirm their status (such as waiting or running). Thus, a visual interface is required for data aggregation systems using mobile agents.

3) Physical Sensor Network: To aggregate the sensor data generated from the sensor networks, the system obviously needs to connect to the physical sensor networks. The sink nodes of the sensor networks collect the sensor data generated from the connected sensors. In addition, the system must accommodate different sensor database schemas, since the sensor networks are managed by different organizations. Thus, a schema-management function is required for managing the physical sensor network.

B. Design

Figure 1 shows the network architecture for our model integrated sensor network. Each sensor network has a gateway, and can connect to the Internet via the gateway. The gateway has rich computing resources, and has no difficulties with battery lifetime as it is connected to an external power source. In the figure, two different types of sensor networks, X-Sensor 1.0 and LiveE!, are included in the integrated sensor network. To integrate different types of sensor networks, we use the sensor network management server, which manages the metadata, such as the ID, location, and database schema of each sensor network. Users can determine these by consulting the sensor network management server. Since the sensor network management server needs to maintain the latest metadata, the metadata are sent to the sensor network management server when the gateways update their metadata. Below, we explain our system design based on the requirements described in the previous subsection.

1) Use of the Mobile Agent System: Various systems to manage mobile agents have been developed. Among these, P2P-based systems are suitable for integrated sensor networks that include many sensor networks. By using a P2P-based mobile-agent system, the system load is not concentrated on the server, and thus it can run the mobile agents effectively. Although our system has a sensor network management server, the load is low since the clients communicate with it only when they start running the system. To facilitate the collection of sensor data generated from integrated sensor networks, we prepare two kinds of special mobile agents: (a) gateway agents which are stationed at the gateways to access sensor networks, and collect sensor data obtained from sink nodes via the gateways; and (b) user agents which are generated by the clients and execute the programs written by users. These latter cannot access the sensor data obtained from the gateway directly, since they do not know the access method for the sensor database of the sink node connected to the gateway. Therefore, they communicate with the gateway agents in order to receive the sensor data.

By stopping the user agents at the gateways, users can check their status. Otherwise, users cannot check their status, since the user agents usually continue to migrate. We defined four user-agent statuses: *READY* means the mobile agents are ready to run programs; *WAIT* means the mobile agents are waiting at each sensor network for the user's commands; *RUNNING* means the mobile agents are executing the user's programs; *FINISH* means the mobile agents have completed the program.

2) Web Browser Interface: To provide visual interfaces, we use web browsers. This is because users do not need to install new software in order to visualize the state of sensor data aggregation, since most of the clients have web browsers. In addition, we can employ various web applications such as maps and databases through the Internet, and exploit these, through web browsers, for the purposes of visualization. By clicking and sending the commands to the indicators for the mobile agent on the map, users can control the user agent based on its status. Furthermore, in the case of an abnormal stop, or the deletion of the mobile agents, the messages are displayed on the web browser.

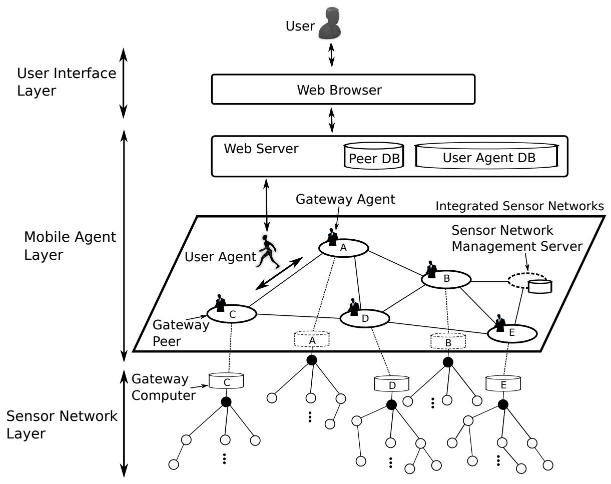

Figure 2 The X-Sensor 2.0 Architecture

3) Integrated Sensor Networks: Since the sensor network management server has the metadata for the sensor networks, we can employ different types of sensor networks that have different sensor database schemas. In our design, the sink nodes must connect to the gateways, and the mobile-agent management system must be installed in the gateways.

IV. Implementation

In this section, we describe our implementation of the data aggregation system using mobile agents. We call the implemented system X-Sensor 2.0. The X-Sensor 2.0 architecture is shown in Figure 2. The architecture has three layers. In Subsection IV-A, we explain the mobile-agent layer. In Subsection IV-B, we explain the user-interface layer; and finally, we explain the sensor-network layer in Subsection IV-C. The X-Sensor 2.0 essentially extends the X-Sensor 1.0 to incorporate the use of mobile agents.

A. Implementation of Mobile Agents

We exploit PIAX ([9]) for managing the mobile agents in the mobile-agent layer. PIAX is a peer-to-peer (P2P)-based mobile-agent system, suitable for aggregating data from integrated sensor networks. The mobile agents are implemented by Java. Our implemented mobile agents have some APIs (Application Programming Interfaces). Users

Table I X-SENSOR 2.0 APIS (PARTIAL LIST)

Class	Main Method	Arguments	Return Value
PeerAccess	getPeerInfo	none	PeerInfo
SQLProcessor	getSensorData	id,site, query	none
	getSensorResult	id	ResultSet
	returnAsText	id	none
	returnAsImage	id, GraphOptions	none
Agent Communication	sendMessage	agentid, message	none
AgentControl	getAgentStatus	none	AgentStatus
	move	peername	none
	stop	none	none

write mobile-agent programs using these APIs. For non-expert users, we can prepare various template programs. The implemented APIs are shown in Table I (X-SENSOR 2.0 APIS). The PeerAccess class provides users access to information about the sensor networks. For example, users can obtain the database schemas and the location of the mobile agents at runtime by using the getPeerInfo method. The SQLProcessor class provides the query processing functions. Users write SQL queries and make the user agents execute them by using the getSensorData method included in the SQLProcessor class. Users can obtain the result by using the getSensorResult method, and then show the result on the web browser by using the returnAsText or returnAsImage methods. The returnAsText method is used to check the results of data aggregation by downloading the text files. The returnAsImage method is used to return the JPEG images to clients. The AgentCommunication class effects communication between the user agents and the gateway agents. For example, when users wish to send messages to a user agent, they use the sendMessage method. Users write the receiveMessage method to receive the messages. The AgentControl class provides the getStatus method and some agent-control methods. By using the getStatus method, users can determine the status of the user agents at runtime, and control them by the move method or the stop method based on their status.

B. Visualization of Sensor Networks

We use web browsers for the user-interface layer, as explained in III-B (2). Figure 3 shows a screenshot of the web browser interface for the X-Sensor 2.0. Users log into the X-Sensor 2.0 system. Then, the map is shown on the web browser. To render the map, we use MSN Virtual Earth [11]. In the figure, we can see sensor networks at Italy, India, China, and Japan. A red square indicates a single sensor network. When the mouse cursor points at the indicator, the detailed information for the sensor network pops up. If there are a number of sensor networks in a narrow area, it may be difficult to recognize each indicator. In this case, the indicators are bundled up in one square indicator. By

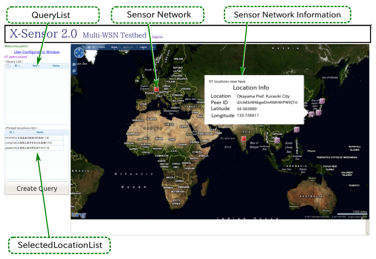

Figure 3 The X-Sensor 2.0 Web Interface

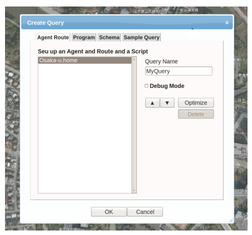

Figure 4 Query Creation Dialogue

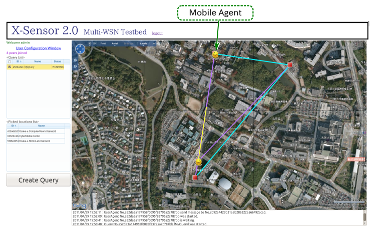

Figure 5 Visualization of Mobile Agents

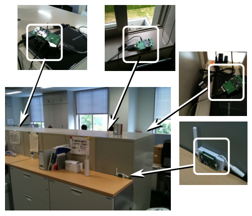

Figure 6 Sensors for X-Sensor 2.0 in Our Laboratory

zooming the map, the bundled indicator is divided into the respective squares. To display the information for each sensor network when users access the web site, the web server accesses the sensor network information from the sensor network management server.

To aggregate the data from integrated sensor networks, users first select the respective sensor networks. The selected sensor networks are listed in the *SelectedLocationList*, which is shown on the left side of the web page. When users click the selected sensor network name on the *SelectedLocationList*, the location is shown on the map. By double-clicking the sensor network name, the sensor network is removed from the *SelectedLocationList*.

After selecting the sensor networks, users push the "Create Query" button to create queries. Here, "query" means a query to aggregate sensor data, including the program for the mobile agents. To answer the queries, the system generates the user agents. Unless the user writes the mobile-agent generation function in the program for the user agent, one query generates one user agent. Figure 4 shows the query creation dialogue. With the *AgentRoute* tab, users enter the query name. Then, users select the agent mode. The agent modes include a normal mode and a debug mode. In the normal mode, the mobile agents do not wait for user commands before migrating through sensor network. Thus, users cannot control the mobile agents until they return to the client. In the debug mode, the mobile agents wait for the 'move' command before they begin to migrate to other sensor networks. The program for the mobile agents is written in the *Program* tab. In the *Schema* tab, users can confirm the metadata for each sensor network. When users finish creating the query, they push the “OK” button. Then, the generated user agent begins to migrate among the sensor networks. The created queries are listed in the *QueryList* located on the top left of the web page.

To facilitate their visualization, the user agents register their events, such as beginning migration or beginning communication with the web server. When the web server receives an event, it sends an asynchronous message to the users’ web browser using Reverse Ajax. When a mobile agent migrates to a peer, the location is shown in the map, as in Figure 5.

The X-Sensor 2.0 can show the aggregation results by means of a graph image, as shown in Figure 7. Users write in the programs how to visualize the results of data aggregation and generate the graph image from query results; then download the results. We explain the detailed program in Subsection IV-D.

Users can check the results of the data aggregation by double-clicking the query name listed in the *QueryList*. In addition, since the mobile agents can return the results at runtime, users can check the intermediate results when needed.

C. Implementation of Integrated Sensor Networks

We need only two steps to add a new sensor network to the X-Sensor 2.0. For example, first, users create a configuration file for the new sensor network. Next, users install and activate the PIAX on the gateway computer, and connect the sensor network to the integrated sensor network. Figure 6 shows the physical sensors that compose the X-Sensor 2.0. We deployed more than 100 sensor nodes in our laboratory.

D. Application Scenario

In this subsection, we describe how to aggregate the sensor data obtained from the X-Sensor 2.0. The X-Sensor 2.0 is available as web site (see [12]). Suppose the case that the user obtains the average temperature from an integrated sensor network. The integrated sensor network consists of three sensor networks, *Osaka-u.NishioLab.Xsensor1*, *Osaka-u.NishioLab.Xsensor2*, and *CyberMedia.Center*. These sensor networks have temperature sensors, and periodically collect the sensor data into their databases. First, the user logs into the system and consults the map. The user selects some of these networks by clicking their indicators shown on the map. Then, the user clicks the “Create Query” button, and chooses a template program for the mobile agents. The template program for this example is shown in Figure 8.

Lines 1 and 2 are required to employ the user agent and the SQL query API respectively. The userProcessing method beginning with line 6 is executed when the mobile agent migrates to other sensor networks. In line 8, the variable *sp* is initialized and manages the query results of each sensor network. In line 9, the mobile agent aggregates the *timestamp* and the temperature

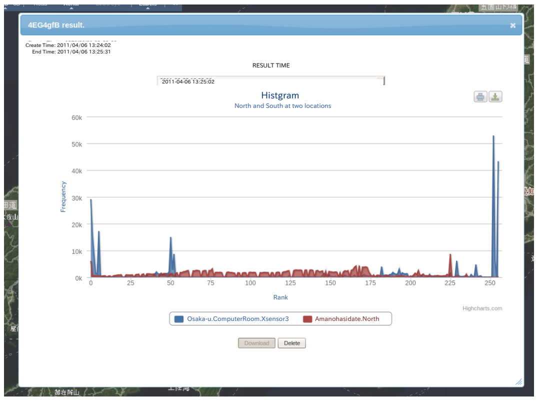

Figure 7 Visualization of Results

```
1  import org.xsensor2.agent.TraverseAgent;
2  import org.xsensor2.dbaccess.SQLProcessor;
3
4
5  public class UserAgent extends TraverseAgent{
6    public void userProcessing() {
7
8      SQLProcessor sp = this.getSQLProcessor();
9      sp.getSensorData(”key1”,”select timestamp,value from temperature
             limit 20”,”Osaka−u.NishioLab.Xsensor1”);
10     sp.getSensorData(”key1”,”select timestamp,value from temperature
             limit 20”,”Osaka−u.NishioLab.Xsensor2”);
11     sp.getSensorData(”key1”,”select timestamp,value from temperature
             limit 20”,”CyberMedia.Center”);
12   }
13
14   public void finalProcessing() {
15
16     SQLProcessor sp = this.getSQLProcessor();
17     sp.getSensorData(”avg”,”SELECT sensortype,AVG(value) FROM key1
             ”);
18     sp.returnAsImage(”avg”);
19
20   }
21 }
```

Figure 8 A Program for User Agents

data as a *value* from the *Osaka-u.NishioLab.Xsenseor1* sensor network. Also, in lines 10 and 11, the mobile agent aggregates the same sensor data obtained from *Osaka-u.NishioLab.Xsensor2* and *CyberMedia.Center*. The finalProcessing method beginning with line 14 is executed when the mobile agent returns to the client. In the method, the program calculates the average of the aggregated sensor data. At this time, *key1* has the query results obtained from the getSensorData method. After writing the program, the user creates the mobile agent, which migrates among the sensor networks. After a while, the mobile agent returns to the client and the result is shown on the web browser.

E. System Evaluation

We measured the network traffic for data aggregation using X-Sensor 2.0 implementation and server client implementation, as a comparison. At the measurement, 5 sensor networks are connected, and the size of each sensor data is 10 bytes. There are over 20 sensors in each sensor network,

Table II THE EVALUATION RESULT

Aggregation Method	Network Traffic	
	Total 500 records	Total 50,000 records
Server Client	24,414 bytes	217,450 bytes
Mobile Agent	45,695 bytes	45,730 bytes

and user submits the query to calculate average value for all sensor data. We created the query and the mobile agent program to get 100 or 10,000 records at each sensor network. The size of the query and the mobile agent program are 618 and 9013 bytes respectively. The network traffic includes the data for mobile agents. The result is shown in Table II. In server-client method, the server aggregates all sensor data. Then, it calculates the average value. In mobile-agent method, the mobile agent calculates the average value while migrating among gateways successively.

V. Discussion

We can see that the mobile agent method can reduce the network traffic compared with the server-client method. The server-client method requires more network traffic than the mobile agent method as the sensor data size increases. However, when the mobile agent can not aggregate sensor data effectively, the network traffic increases. Therefore, the effectiveness of using mobile agents depends on the application of data aggregation.

X-Sensor 2.0 is suitable for sensor data aggregation using mobile agents, since the system is designed to make the mobile agents aggregate the sensor data. Other mobile-agent systems such as AgentTeamwork or the original PIAX are designed for, and can be applied to, various mobile-agent applications. However, a mobile-agent system customized for integrated sensor networks offers users easy and efficient aggregation of the sensor data generated from integrated sensor networks, since the mobile agent migrates among multiple sensor networks. A further merit may be seen in the X-Sensor 2.0's ability to visualize the locations of sensor networks and the mobile agents. Therefore, it is easy to determine the status of the mobile agents. Regarding the fault tolerance for X-Sensor 2.0, by programming the mobile agents so that they can find unusual response from sensors, we can make the system be tolerant for troubles.

On the other hand, one of the demerits of the X-Sensor 2.0 is that it is hard for non-experts users to write the agent programs. We can solve this problem by improving the script language for programming the mobile agents, and supporting more intuitive GUI. In addition, since users have flexibility in writing mobile-agent programs, they can create malicious agents such as computer viruses. However, we can detect these by checking the user-written mobile-agent programs at the web server.

VI. Conclusion

In this paper, we described the design and implementation of the X-Sensor 2.0, which is a data aggregation system using mobile agents on integrated sensor networks. In the system, the mobile agents migrate among the sensor networks, executing operations on the sensor data. Since the mobile agents aggregate only necessary sensor data, network traffic can be reduced.

In the future, we will define a more intuitive script language for programming the mobile agents.

Acknowledgment

This research was supported by collaborative research of NICT·Osaka University (research and development, validation of integrated management technique for heterogeneous, wide-area sensor networks) and Grants-in-Aid for Scientific Research (S) numbered 21220002.

References

[1] S. Michel, A. Salehi, L. Luo, N. Dawes, K. Aberer, G. Barrenetxea, M. Bavay, A. Kansal, K. A. Kumar, S. Nath, M. Parlange, S. Tansley, C. van Ingen, F. Zhao, and Y. Zhou, "Environmental monitoring 2.0.", In Proc. of the International Conference on Data Engineering (ICDE 2009), pp. 1507-1510, 2009.

[2] P. Gibbons, B. Karp, Y. Ke, S. Nath, and S.n Sechan, "IrisNet: An Architecture for a Worldwide Sensor Web.", IEEE Pervasive Computing, Vol. 2, No. 4, pp. 22-33, 2003.

[3] Live E!: Live environmental information of the earth, http://www.live-e.org/en, Sep 3, 2011.

[4] A. Kanzaki, T. Hara, Y. Ishi, T. Yoshihisa, Y. Teranishi, and S. Shimojo, "X-Sensor: Wireless Sensor Network Testbed Integrating Multiple Networks.", Wireless Sensor Network Technologies for the Information Explosion Era Studies in Computational Intelligence, Vol. 278, pp. 249-271, 2010.

[5] X-Sensor Multi-WSN Testbed, http://www1.x-sensor.org, Sep 3, 2011.

[6] S. Nath, J. Liu, and F. Zhao, "Sensormap for wide-area sensor webs.", IEEE Computer, Vol. 40, No. 7, pp. 90-93, 2008.

[7] K. Aberer, M. Hauswirth, and A Salehi, "Infrastructure for Data Processing in Large-Scale Interconnected Sensor Networks.", In Proc. of the International Conference on Mobile Data Management (MDM 2007), pp. 198-205, 2007.

[8] M. Fukuda, C. Ngo, E. Mak, and J. Morisaki, "Resource management and monitoring in AgentTeamwork grid computing middleware.", In Proc. of the IEEE Pacific Rim Conference on Communications, Computers, and Signal Processing (PacRim 2007), pp. 145-148, 2007.

[9] Y. Teranishi, "PIAX: Toward a Framework for Sensor Overlay Network.", In Proc. of the International IEEE Consumer Communications and Networking Conference Workshop on Dependable and Sustainable Peer-to-Peer Systems (CCNC 2009), pp. 1-5, 2009.

[10] H. Qi, S. S. Iyengar, and K. Chakrabarty, "Distributed Multi-Resolution Data Integration Using Mobile Agents.", In Proc. of the IEEE Aerospace Conference, Vol. 3, pp. 1133-1141, 2001.

[11] MSN Virtual Earth, http://virtualearth.msn.com, Sep 3, 2011.

[12] X-Sensor 2.0, http://www2.x-sensor.org, Sep 20, 2011.