5

An Agent-Based System for Sensor Cloud Management

Yu-Cheng Chou, Bo-Shiun Huang and Bo-Jia Peng
Chung Yuan Christian University
Taiwan

1. Introduction

An embedded sensor network is a network of embedded computers deployed in the physical world that interacts with the environment. Each embedded computer, commonly referred to as a sensor node, is a physically small and relatively inexpensive computer that has one or more sensors. These sensor nodes are often networked, allowing them to communicate and cooperate with each other to monitor the environment[1]. In recent years, embedded sensor networks have been gaining increasing attention, both from the academia and industry, because of their potential to be a novel and practical solution across multiple areas such as industrial automation, asset management, environmental monitoring, transportation business, and healthcare. Typically, an embedded sensor network is controlled by its own applications that can access the sensor nodes within the network. On the other hand, the sensor nodes cannot be easily accessed by applications outside of the network. Moreover, even within the same network, different applications might encounter a race condition when they are trying to access a sensor node simultaneously. Existing research works of embedded sensor networks focus on data processing[2], routing[3, 4], power management[5], clock synchronization[6-8], localization[9, 10], operating system[11, 12], and programming[13, 14]. However, not much research has been done with a focus on the management of sensor nodes.

In the past few years, Cloud computing[15] has emerged as a new computing paradigm to provide reliable resources, software, and data on demand. As for resources, essentially, Cloud computing services provide users with virtual servers. Users can utilize virtual servers without concerning about their locations and specifications. With such an inspiration, this project proposes a system, Sensor Agent Cloud, where users can access the sensor nodes without worrying about their locations and detailed specifications.

Sensor Agent Cloud virtualizes a physical sensor node as a virtual "sensor agent". Users can use and control sensor agents with standard functions. Dynamically grouped sensor agents are provisioned in response to the users' requests. Users can destroy their sensor agents when they are not needed. Monitoring sensor agents is used to maintain the quality of service. Sensor Agent Cloud also provides a user interface for registering / deleting sensor nodes, requesting for provisioning / destroying sensor agents, controlling / monitoring sensor agents, and registering / deleting users.

Each sensor agent operates on behalf of its user. The mandatory coordination of these sensor agents, which might belong to different users and try to access the same sensor node, is related to the system management. Therefore, Sensor Agent Cloud shall be self-managing. In other words, Sensor Agent Cloud shall be an autonomic system. For a system to be self-managing, there must be an automated method to collect the details of the system, to analyze the details to determine whether any changes need to be made, to create a plan that specifies the necessary changes, and to perform the created plan[16, 17]. An autonomic system shall possess one of the following four autonomic properties, recognized as the four fundamental areas of self-management:

- *Self-configuring* property that enables the system to adapt to unpredictable conditions by automatically changing its configuration, such as adding or removing new components or resources, or installing software changes without disrupting the services provided by itself.
- *Self-healing* property that can prevent and recover from failure by automatically discovering, diagnosing, circumventing, and recovering from issues that might cause service disruptions.
- *Self-optimizing* property that enables the system to continuously tune itself, that is, proactively to improve on existing processes and reactively in response to environmental conditions.
- *Self-protecting* property that detects, identifies, and defends against viruses, unauthorized access, and denial-of-service attacks. Self-protection also can include the ability to protect itself from physical harm.

Therefore, it is expected that our proposed Sensor Agent Cloud, enabling users to easily access various kinds of sensor nodes residing in different embedded sensor networks, can enhance the applicability and usability of embedded sensor networks in many application areas.

The remainder of the chapter will be organized as follows. Section 2 introduces related works in this regard. Section 3 describes design considerations of Senor Agent Cloud. System architecture and implementation will be presented in Sections 4 and 5, respectively. Section 6 illustrates a prototype sensor node. A proof-of-concept, real-world application will be used in Section 7 to validate the Sensor Agent Cloud. Conclusions and future work will be discussed in Section 8.

2. Related works

There have been a few studies focusing on the management of physical sensors. OGC (Open Geospatial Consortium)[18] defined a language called Sensor Modeling Language (SensorML)[19] that provides standard models and an XML encoding for physical sensors' description and measurement processes. SensorML can represent the metadata for any physical sensors (such as the type, the location, and the accuracy).

Although there are many kinds of physical sensors, no applications need to use all of them. Each application only needs sufficient physical sensors that meet its requirements. A publish / subscribe mechanism is used to select physical sensors[20, 21]. When there are multiple sensor networks, each sensor network publishes sensor data and metadata that describe the type physical sensors. Each application subscribes to one or more sensor networks in order

to receive a real-time data stream from the physical sensors. Such publish / subscribe mechanism allows each application to select only the type of physical sensors where it collects data.

Users shall check whether the physical sensors are available and detect physical sensors' faults in order to maintain the quality of data coming from physical sensors. FIND[22] provides a novel approach to detect physical sensors with faulty data. FIND ranks the physical sensors based on their sensing readings and their physical distances from an event. FIND considers a physical sensor to be faulty when there is a significant mismatch between the sensor data rank and the distance rank.

3. Design considerations of sensor agent cloud

An overview of Sensor Agent Cloud is shown in Figure 1. The bottom layer of Sensor Agent Cloud is the sensor node layer that contains various kinds of physical sensor nodes. The next layer is the sensor agent group layer that contains sensor agent groups consisting of virtual sensor agents. The next layer is the regulation layer that is responsible for monitoring, provisioning, and control of sensor agent groups. The top layer is the user interface layer through which users can access their sensor agent groups. The user interface includes functions and a GUI web page. A sensor agent group requested by a user is automatically provisioned by Sensor Agent Cloud. Users can control and monitor their sensor agents via customized programs or a Web browser.

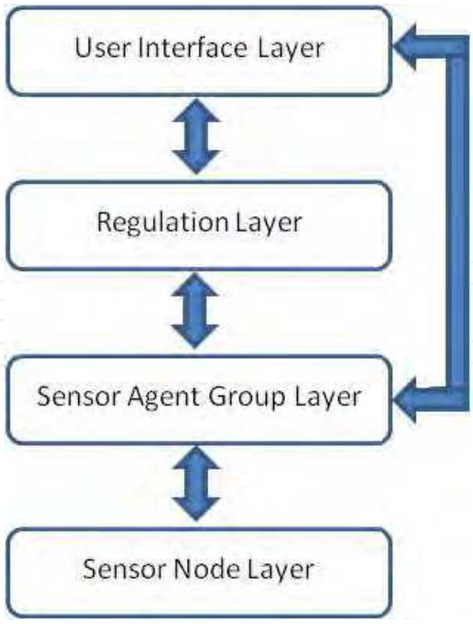


Fig. 1. Overview of Sensor Agent Cloud.

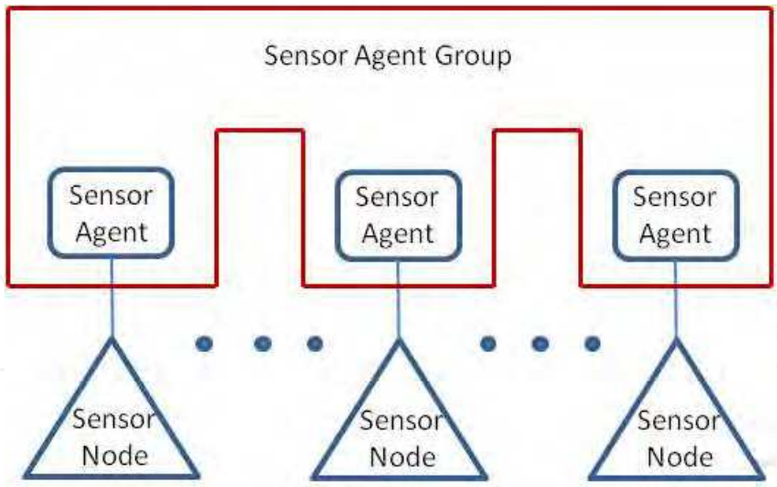


Fig. 2. Relationships among sensor node, sensor agent, and sensor agent group.

The design considerations of Sensor Agent Cloud are illustrated as follows.

1. **Virtualization**: There are various kinds of physical sensor nodes scattered in different locations and networks. The main idea of this project is that, through sensor agent groups, users are able to use physical sensor nodes without worrying about their locations and specifications. Figure 2 describes the relationships among sensor agent groups, sensor agents, and physical sensor nodes. Each sensor agent corresponds to one sensor node. A sensor agent group is created from one or more sensor agents. Users can create sensor agent groups and use sensor agents included in the groups as if they own the physical sensor nodes. For example, they can activate or inactivate their sensor agents, check their status, and set the data collection frequency. The advantage of adopting virtualization is to avoid race conditions that would otherwise occur when multiple users are trying to access or control the same physical sensor node.
2. **Standardization**: Different kinds of sensor nodes have different specifications. Each sensor node provides its own functions for control and data collection. A standard mechanism allows users to access sensor nodes without considering about their different specifications. A set of standard functions are defined for sensor agents, so that users can access sensor agents through standardized functions. A translation mechanism is designed to translate standard functions for the sensor agents into specific functions for different kinds of sensor nodes.

   Standardization also needs to reflect from the aspects of programming and communication languages. Sensor Agent Cloud adopts C programming language and XML-based agent communication language specified in the IEEE FIPA (Foundation for Intelligent Physical Agents) standard. C is an internationally standardized language with a large user base. C has both high-level and low-level functionalities to accommodate the diversity of scientific and engineering applications. Therefore, a huge amount of existing C resources can be used to define a set of standard functions in Sensor Agent Cloud for sensor agent access.

   Since Sensor Agent Cloud employs agents, a standard for agent technology shall be adopted as well. The IEEE FIPA [23] is an international agent standard. The majority of

actively maintained Java-based or C-based multi-agent platforms, such as JADE and Mobile-C[24, 25], are IEEE FIPA compliant platforms. Additionally, members of the IEEE FIPA standard include researchers and experts from many academic institutions and industrial companies. Therefore, by adopting the IEEE FIPA standard, Sensor Agent Cloud is able to interoperate with a growing number of FIPA compliant agent platforms.

3. **Automation**: As mentioned in Section 1, for a system to be self-managing, automation is mandatory to keep the human interference to the minimum. Automation therefore improves the service delivery time and reduces the cost. Sensor Agent Cloud provides templates for sensor agents and sensor agent groups. When a user completes a template for a sensor agent group, Sensor Agent Cloud dynamically and automatically provisions the sensor agent(s) in the sensor agent group specified in the template. Sensor Agent Cloud supports on-demand service delivery as well as the full lifecycle of service delivery, starting from the registration of physical sensor nodes, creation of templates for senor agents, request, provision, use, and release for sensor agents, and deregistration (removal) of physical sensor nodes. These services are automatic and delivered without human interference.
4. **Monitoring**: An application is unable to perform properly and correctly when necessary sensor data cannot be obtained from sensor agents. Therefore, an application owner or user shall be able to check whether or not the sensor agents are still available and monitor the status of the sensor agents for maintaining the quality of service. Users can perform the checking and monitoring operations through Sensor Agent Cloud's monitoring mechanism.
5. **Clustering**: There might be a variety of different physical sensor nodes existing in a network. However, an application is unlikely to use all of them. An application essentially uses certain types of sensor nodes. Sensor Agent Cloud provides virtual sensor agent groups. Users can control each virtual sensor agent group. For instance, a user can set the access control and the frequency of data collection for a virtual sensor group. Sensor Agent Cloud provides templates for typical sensor agent groups. But a user can also create new sensor agent groups by incorporating his/her desired sensor agents.
6. **Participant Role**: Three different roles for participants are defined in Sensor Agent Cloud.
   a. *Sensor Node Owner*: A sensor node owner possesses physical sensor nodes. A sensor node owner allows others to use his/her sensor nodes through Sensor Agent Cloud. A sensor node owner registers his/her sensor nodes along with their properties to Sensor Agent Cloud. On the other hand, a sensor node owner deregisters the sensor nodes when he/she is not willing to share them anymore.
   b. *Sensor Agent Cloud Administrator*: A Sensor Agent Cloud Administrator manages the Sensor Agent Cloud system. An administrator manages the required computing resources for sensor agents, monitoring mechanism, and user interface. An administrator also needs to provide templates for sensor agents and some typical sensor agent groups.
   c. *End-User*: An end-user has one or more applications that use the sensor data. Through templates, an end-user requests the use of sensor agent groups that satisfy the requirements of his/her applications. Basic and typical templates are provided

by Sensor Agent Cloud administrators. An end-user can also create a new template for a sensor agent group by incorporating multiple templates of sensor agents or by modifying the existing template of a sensor agent group. End-users can share their own templates among other end users. An end-user can control his/her sensor agents through programs with standard functions or via a Web interface provided by Sensor Agent Cloud. An end-user can monitor the status of sensor agents. When sensor agents become not needed, an end-user can release them.

4. System architecture of Sensor Agent Cloud

The system architecture of Sensor Agent Cloud is shown in Figure 4. The functions of main components are described as follows.

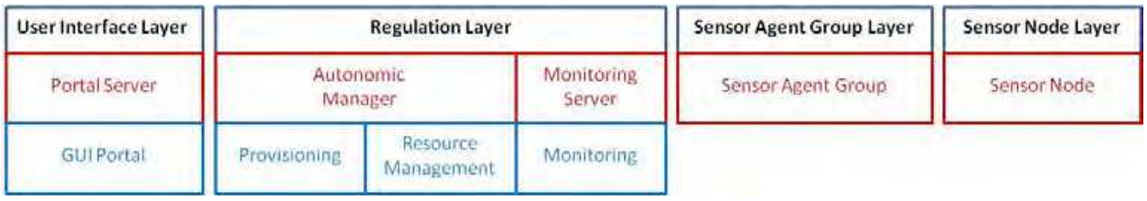


Fig. 3. System architecture of Sensor Agent Cloud.

1. Portal server: When a user logs into the portal from a Web browser, the user's role - sensor node owner, Sensor Agent Cloud administrator, or end-user, determines the operations available to the user. For an end-user, the portal server shows menus for logging in, logging out, requesting for provisioning or destroying sensor agent groups, monitoring sensor agents, controlling sensor agents, and creating templates for sensor agent groups. For a sensor node owner, the portal server gives menus for logging in, logging out, registering sensor nodes, and deleting sensor nodes. For a Sensor Agent Cloud administrator, the portal server gives menus for creating, modifying, and deleting templates for sensor agents and sensor agent groups. Other menus shown to a Sensor Agent Cloud are used to register or delete virtual servers, to manage end-users and sensor node owners, and to check the status of virtual servers. All of the menus for end-users and sensor node owners are available to a Sensor Agent Cloud administrator.
2. Autonomic Manager: The autonomic manager provisions sensor agent groups for requests from the portal server. It contains a workflow engine and predefined workflows. It executes the workflows in a proper order. First, it checks and reserves the computing resource pool when it receives a request for provisioning. It retrieves the templates for sensor agents and sensor agent groups, and then provisions the requested sensor agent groups including sensor agents on the existing or a new virtual server. It also provides virtual servers with monitoring agents. After provisioning, the autonomic manager updates the information of the sensor agent groups.
3. Sensor Agent Group: A sensor agent group is automatically provisioned on a virtual server by the autonomic manager. Each sensor agent group is possessed by an end-user and contains one or more sensor agents. End-users can control the sensor agents. For instance, they can activate or inactivate their sensor agents, set their frequency of data collection, and check their status. Sensor agent groups can be controlled directly or via a Web browser.

4. Monitoring Server: The monitoring server receives the data about sensor agents from the monitoring agents residing in the virtual servers. It stores the received data in a database. The monitoring information for sensor agents can be accessed through a Web browser. Sensor Agent Cloud administrators are able to monitor the status of the monitoring servers as well.

5. Implementation blueprint of Sensor Agent Cloud

Figure 4 shows the implementation blueprint of Sensor Agent Cloud. As shown in Figure 4, the autonomic manager is planned to be implemented as a portal server as well.

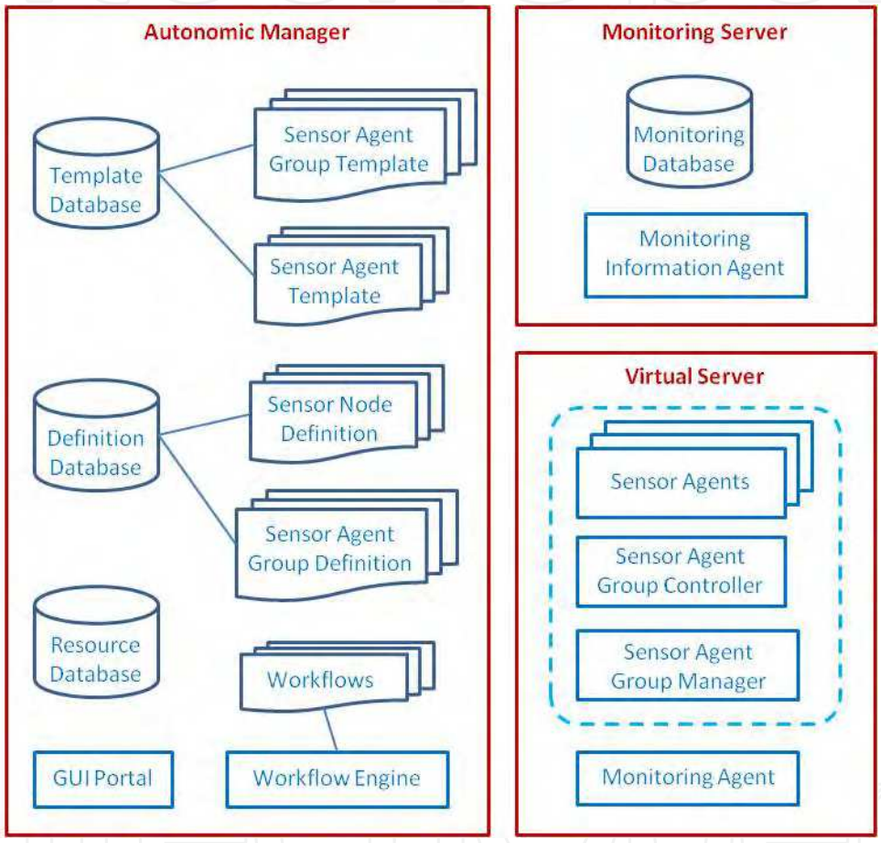


Fig. 4. Implementation blueprint of Sensor Agent Cloud.

There are three databases maintained by the autonomic manager. The definition database stores definitions of the sensor nodes and the provisioned sensor agent groups. Definitions of sensor nodes describe sensor nodes' properties, such as the owner's ID and the type of sensor data. Definitions of sensor agent groups describe the provisioned sensor agent group, such as sensor agent group's ID, end-user's ID, and virtual server's ID. The template database stores the templates for the sensor agents and sensor agent groups. A sensor agent template contains C functions and the data mapping rule. A sensor agent group's template defines the links to templates of included sensor agents and the creator's ID. The workflow engine contains the workflows for each task, such as provisioning sensor agent groups, registering sensor nodes, and controlling sensor agents. The resource database stores the IT

resource pool's definition that describes the data regarding the autonomic manager, monitoring server, and virtual servers, such as IP address, host name, specifications, and usage. The monitoring information agent receives the data from monitoring agents residing in virtual servers and stores in the database. Each virtual server has a monitoring agent, a sensor agent group manager, a sensor agent group controller, and one or more sensor agents. The sensor agent group controller has methods for controlling sensor agents, such as activating or inactivating them, and specifying their frequency of data collection. The methods are called by the workflow for controlling sensor agents. The end-user's application can also call the methods directly. Each sensor agent contains a standard method for accessing its corresponding physical sensor node's specification. An application can get data originating from sensor nodes through standard methods contained in sensor agents.

The flow for provisioning a sensor agent group is illustrated as follows.

1. An end-user logs in the GUI portal on a Web browser.
2. The autonomic server gets the list of templates for sensor agents and sensor agent groups from the Template Database and shows the list to the user.
3. The user requests for selecting and provisioning a sensor agent group.
4. The autonomic manager calls the workflow engine with the user's ID and the template's ID for the requested sensor agent group.
5. The workflow engine executes the workflow for provisioning the sensor agent group.
6. The workflow updates IT resource pool for reservation and gets the virtual server's information.
7. The workflow gets the sensor agent group's template by ID and the sensor agents' templates from the links inside the sensor agent group's template.
8. The workflow creates and sends sensor agents, a sensor agent group controller, and a sensor agent group manager to the virtual server.
9. The workflow adds the definition of the new sensor agent group to the Definition database.
10. The workflow informs the user of the completion of provisioning.

6. A Prototype sensor node

The fundamental building block of the proposed Sensor Agent Cloud is a sensor node that supports agents, C language, and IEEE FIPA standard. Thus, a prototype sensor node supporting those three features is presented in this section.

With regard to software, Mobile-C[24, 25], an embeddable multi-agent agent platform supporting C/C++ mobile agents, is employed in Sensor Agent Cloud as the sensor node agency and client agency, as shown in Figure 5. Among the Mobile-C modules shown in Figure 5, by default, the three FIPA mandatory modules, the Agent Management System (AMS), Agent Communication Channel (ACC), and Directory Facilitator (DF), are initialized when Mobile-C is started. The AMS is related to the creation, registration, execution, migration, persistence, and termination of a mobile agent. The ACC is related to the inter-agency mobile agent transport and inter-agent communication. The DF is related to yellow page activities. The Agent Security Manager (ASM) is responsible for maintaining the security policies for an agency. Here, mobile agents are assumed authorized mobile agents. Thus, the ASM is not required to be initialized in Mobile-C. By default, the ASM is not initialized when Mobile-C is started.

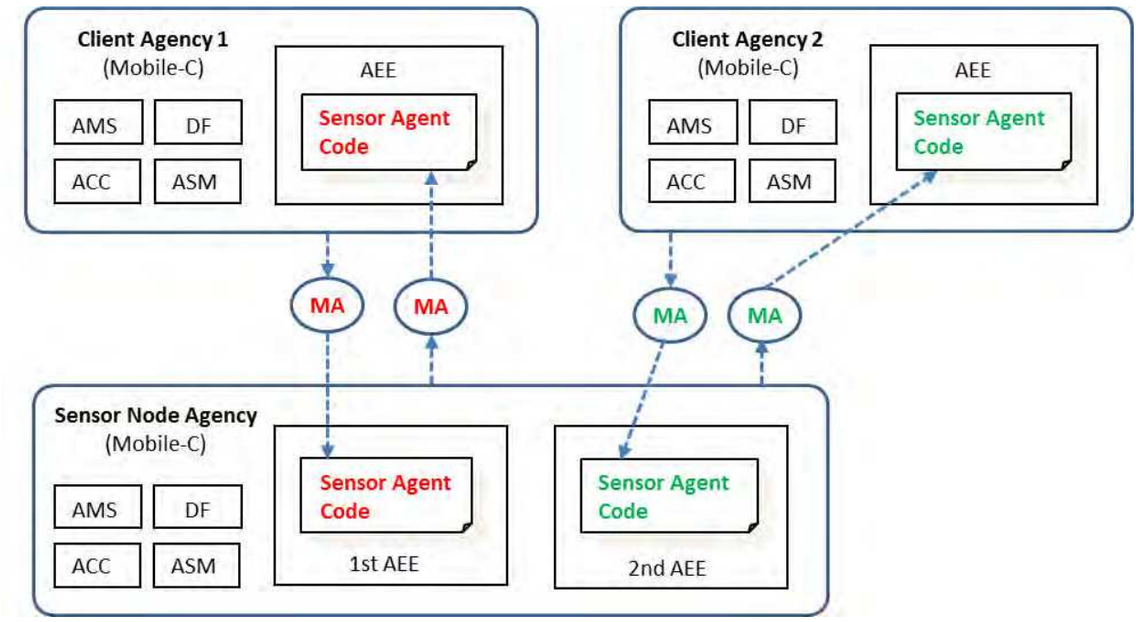


Fig. 5. Sensor node agency and client agency in Sensor Agent Cloud.

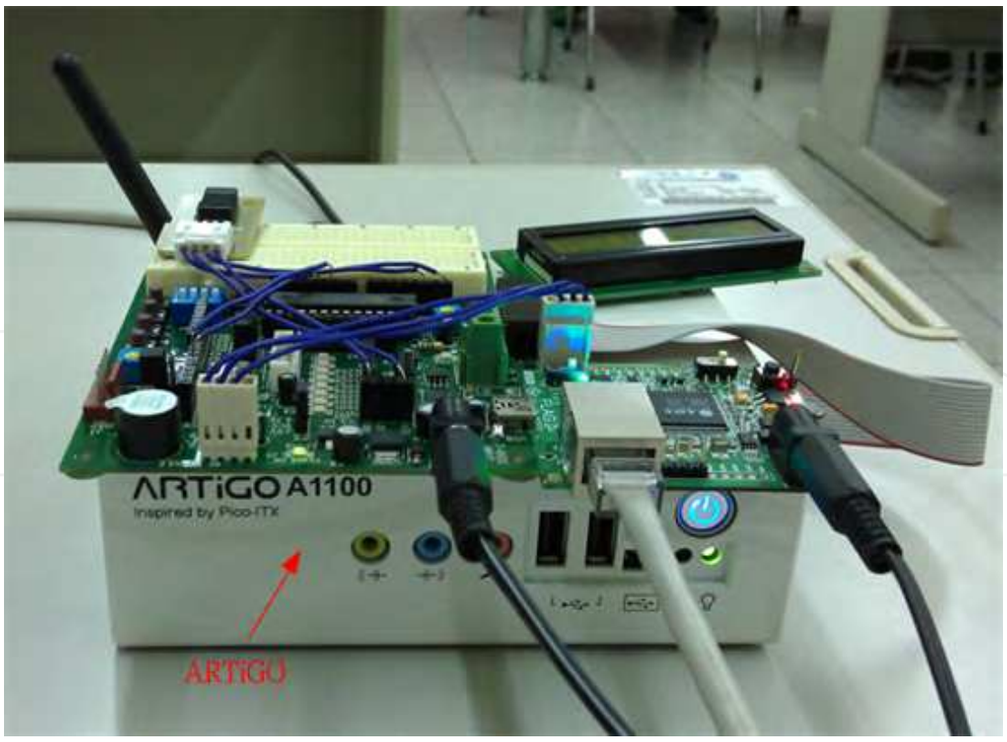


Fig. 6. VIA ARTiGO A1100 computer of the prototype sensor node.

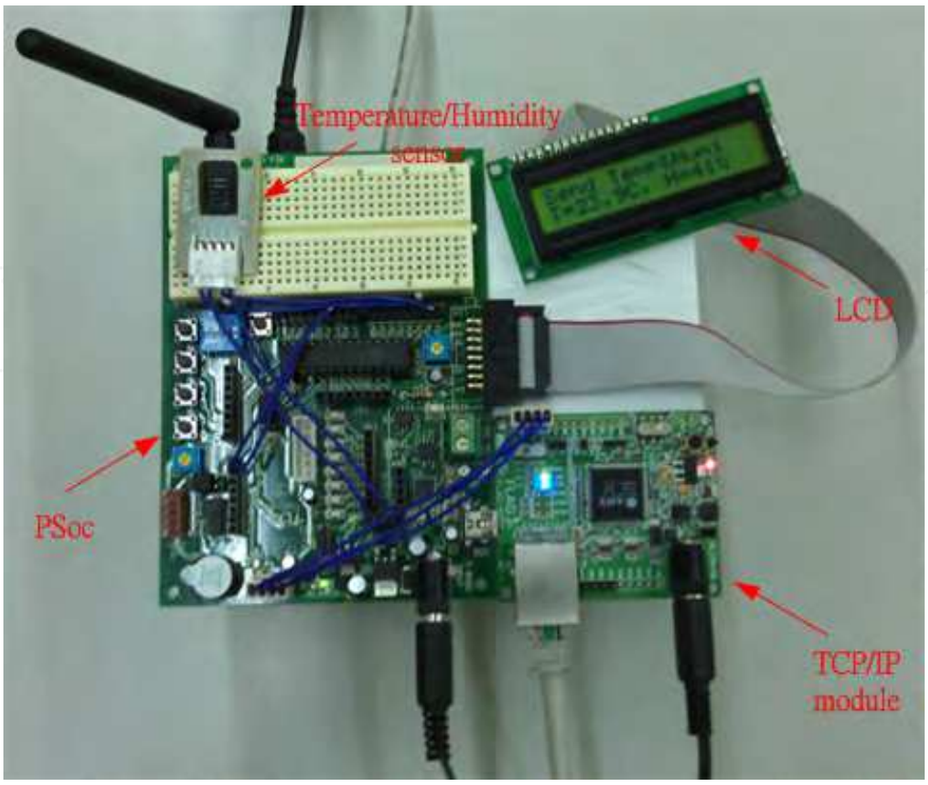


Fig. 7. FLAG PSoC development system of the prototype sensor node.

A mobile agent has an XML structure where a C/C++ sensor agent code is embedded[26]. As shown in Figure 5, after a mobile agent is created at a client agency, it will migrate to a specified sensor node agency. When a mobile agent arrives at a sensor node agency, an Agent Execution Engine (AEE) will be automatically created to run the sensor agent code. Once the execution of the sensor agent code is finished, the result will be carried by the original mobile agent back to its home client agency. Similarly, when such a mobile agent arrives at its home client agency, an AEE will be created to run the sensor agent code, which typically displays the result required by a user. Ch[27], a C/C++ interpreter, is the AEE of Mobile-C.

With regard to hardware, the prototype sensor node consists of a VIA ARTiGo A1100[28] and a FLAG Programmable System on Chip (PSoC) development system[29], as shown in Figures 6 and 7. The VIA ARTiGO A1100 has a size of 14.6 cm x 9.9 cm x 5.2 cm, and has key features including a 1.2GHz VIA Nano processor, hardware accelerated video decoding, multiple I/O ports, and Gigabit Ethernet and wireless networking. The FLAG PSoC development system is composed of a PSoC development board (FLAG-1605A) with a Cypress programmable chip (CY8C29466-24PXI), a TCP/IP network module (FLAG-N001), an LCD module, and a temperature and humidity sensor module (FLAG-1613A).

7. Dynamic sensor node data retrieval

This section uses an application of dynamic sensor node data retrieval to validate the fundamental building block of Sensor Agent Cloud, the sensor node, in terms of its ability to execute sensor agent codes to obtain user-desired information.

Figure 8 shows the diagram for the dynamic sensor node data retrieval application. The communication between the PSoC development board and TCP/IP network module is through a Universal Asynchronous Receiver/Transmitter (UART). The Peer-to-Peer (P2P) communication is the communication method between the TCP/IP network module and ARTiGO computer. Moreover, the wireless network communication is the communication method between the ARTiGO computer and a remote desktop computer.

In the PSoC development board, there is a sensor data transmitter program that continuously waits for three seconds, reads the real-time temperature and humidity data from the sensor module, displays the data on the LCD module, and sends the data to the ARTiGO computer through the TCP/IP network module, as shown in Figure 9.

A sensor data receiver program and a sensor node agency are running in the ARTiGO computer. The sensor data receiver program continuously receives the real-time data from the PSoC development board and saves the data into a database, as shown in Figure 10. Here, the database is implemented as a text file for simplicity's sake.

In the sensor node agency, an AEE will be created to run the sensor agent code of an arrived mobile agent. Such a mobile agent has two tasks, each of which corresponds to a sensor agent code. The first sensor agent code will be executed on the ARTiGO computer, which is to access the database and perform calculations on the data to obtain the desired result, the average temperature and humidity, as shown in Figure 11. The result will be carried by the original mobile agent back to its source location, the client agency, where an AEE will be created to run the second sensor agent code to display the average temperature and humidity, as shown in Figure 12.

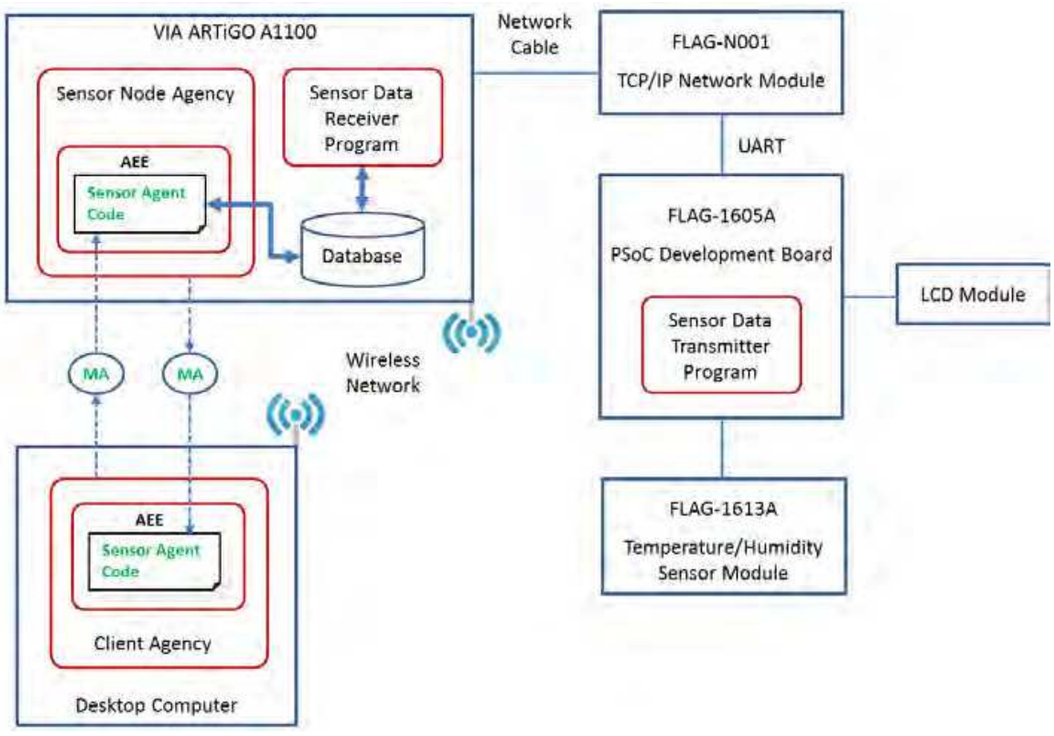


Fig. 8. Dynamic sensor node data retrieval application diagram.

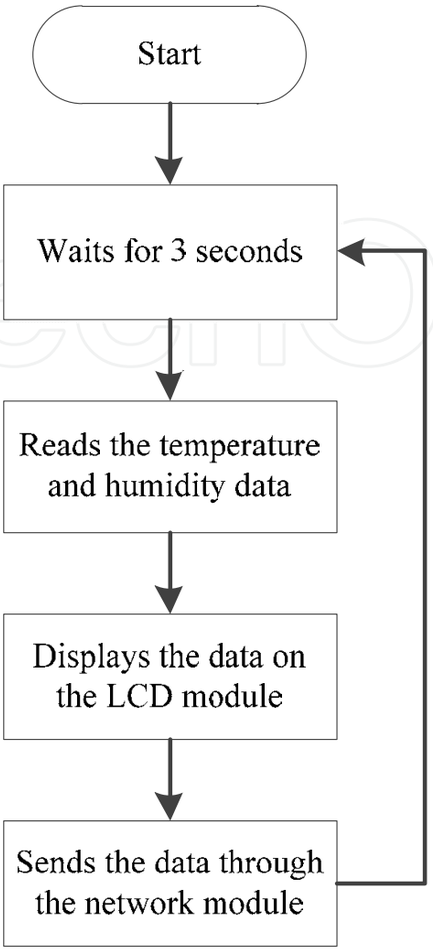


Fig. 9. Simplified flowchart of the sensor data transmitter program.

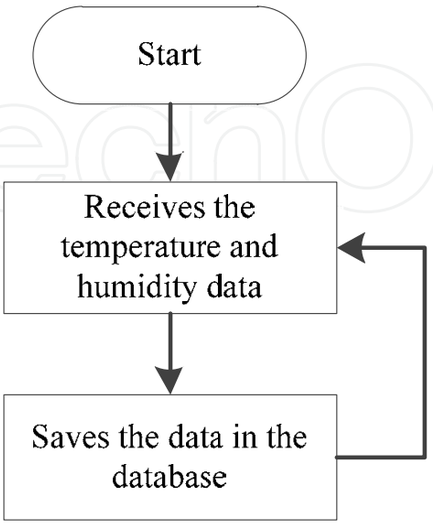


Fig. 10. Simplified flowchart of the sensor data receiver program.

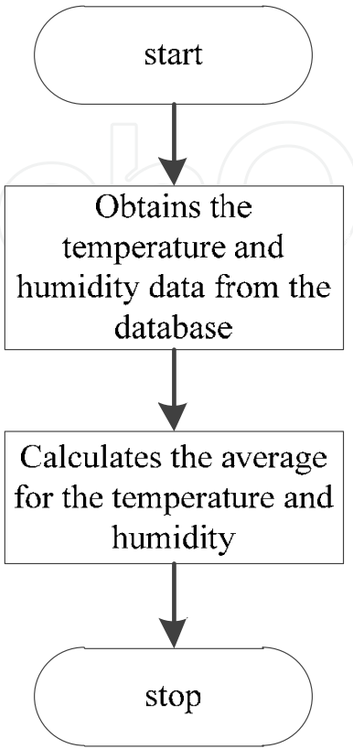


Fig. 11. Simplified flowchart of the 1st sensor agent code carried by the mobile agent.

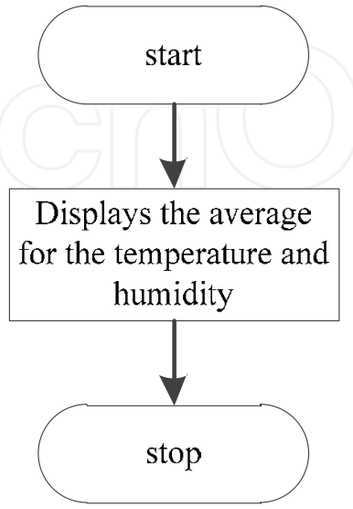


Fig. 12. Simplified flowchart of the 2nd sensor agent code carried by the mobile agent.

Fig. 13. Screenshot of the sensor node agency, server.exe.

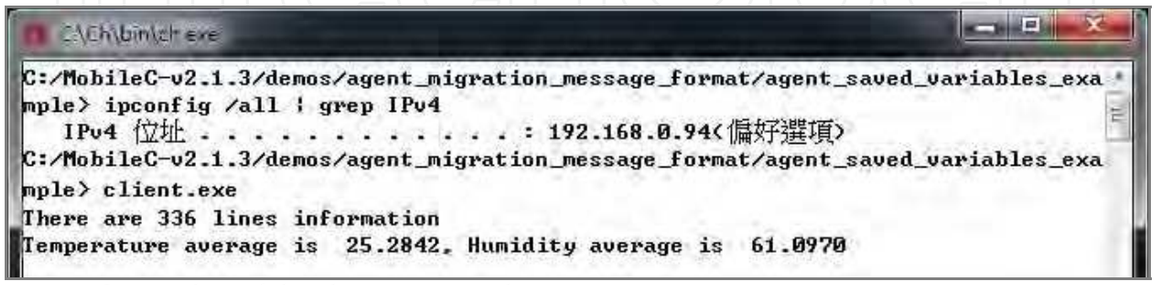


Fig. 14. Screenshot of the client agency, client.exe.

Figures 13 and 14 illustrate the screenshots for the sensor node agency and client agency, respectively. The IP address of the VIA ARTiGO A1100 where the sensor node agency is running is 192.168.0.100, as shown in Figure 13. The IP address of the remote desktop computer where the client agency is running is 192.168.0.94, as shown in Figure 13. Moreover, there are totally 336 lines of information, each of which corresponds to one set of temperature and humidity data. The average temperature and humidity are 25.2842 °C and 61.097%, respectively.

8. Conclusion and future work

This paper presents the architecture and implementation blueprint for a cloud-based autonomic system for sensor nodes management, called Sensor Agent Cloud. The implementation of a prototype sensor node for the Sensor Agent Cloud is also presented. An application of dynamic sensor node data retrieval is used to validate the Sensor Agent Cloud's building block, the sensor node, in terms of its ability to execute sensor agent codes to obtain user-desired information. However, the entire system is still under development. Therefore, the future work is to fully implement the Sensor Agent Cloud and compare it with other representative sensor-cloud systems.

9. Acknowledgment

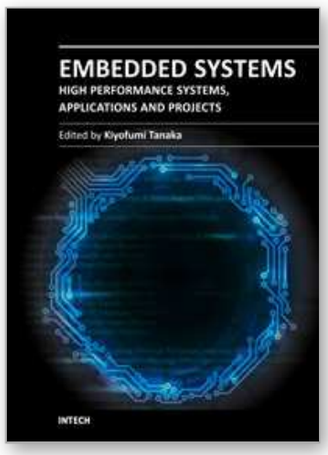
This research is supported by the National Science Council, Taiwan, under grant NSC 100-2221-E-033-080.

10. References

[1] J. Heidemann and R. Govindan, "An Overview of Embedded Sensor Networks," USC/Information Sciences Institute, 2004.

[2] S. R. Madden and M. J. Franklin, "Fjording the Steam: An Architecture for Queries Over Streaming Sensor Data," in The 18th International Conference on Data Engineering, 2002.
[3] M. H. Alizai, O. Landsiedel, J. A. B. Link, S. Goetz, and K. Wehrle, "Bursty Traffic over Bursty Links," 2009.
[4] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection Tree Protocol," in The 7th ACM Conference on Embedded Networked Sensor Systems, 2009.
[5] R. Katsuma, Y. Murata, N. Shibata, K. Yasumoto, and M. Ito, "Extending k-Coverage Lifetime of Wireless Sensor Networks Using Mobile Sensor Nodes," in The 5th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, 2009.
[6] J. Koo, R. K. Panta, S. Bagchi, and L. Montestruque, "A Tale of Two Synchronizing Clocks," in The 7th ACM Conference on Embedded Networked Sensor Systems, 2009.
[7] C. Lenzen, P. Sommer, and R. Wattenhofer, "Optimal Clock Synchronization in Networks," in The 7th ACM Conference on Embedded Networked Sensor Systems, 2009.
[8] A. Rowe, V. Gupta, and R. Rajkumar, "Low-Power Clock Synchronization using Electromagnetic Energy Radiating from AC Power Lines," in The 7th ACM Conference on Embedded Networked Sensor Systems, 2009.
[9] K. Matsumoto, R. Katsuma, N. Shibata, K. Yasumoto, and M. Ito, "Extended Abstract: Minimizing Localization Cost with Mobile Anchor in Underwater Sensor Networks," in The Fourth ACM International Workshop on UnderWater Networks, 2009.
[10] Z. Zhong and T. He, "Achieving Range-Free Localization Beyond Connectivity," in The 7th ACM Conference on Embedded Networked Sensor Systems, 2009.
[11] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System Architecture Directions for Networked Sensors," in International Conference on Architectural Support for Programming Languages and Operating Systems, 2000.
[12] K. Klues, C. Liang, J. Paek, R. Musaloiu-E, P. Levis, A. Terzis, and R. Govindan, "TOSThreads: Thread-Safe and Non-Invasive Preemption in TinyOS," in The 7th ACM Conference on Embedded Networked Sensor Systems, 2009.
[13] J. S. Miller, P. Dinda, and R. Dick, "Evaluating a BASIC Approach to Sensor Network Node Programming," in The 7th ACM Conference on Embedded Networked Sensor Systems, 2009.
[14] T. I. Sookoor, T. W. Hnat, P. Hooimeijer, W. Weimer, and K. Whitehouse, "Macrodebugging: Global Views of Distributed Program Execution," in The 7th ACM Conference on Embedded Networked Sensor Systems, 2009.
[15] A. Weiss, "Computing in the Clouds," netWorker, vol. 11, pp. 16-25, 2007.
[16] IBM, "IBM Autonomic Computing White Paper - An Architectural Blueprint for Autonomic Computing," 2005.
[17] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," IEEE Computer, vol. 36, pp. 41-50, 2003.
[18] "OGC: Open Geospatial Consortium," http://www.opengeospatial.org/.
[19] "SensorML: Sensor Modeling Language," http://vast.uah.edu/SensorML/.

[20] M. Gaynor, M. Welsh, S. Moulton, A. Rowan, E. LaCombe, and J. Wynne, "Integrating Wireless Sensor Networks with the Grid," IEEE Internet Computing, 2004.
[21] J. Shneidman, P. Pietzuch, J. Ledlie, M. Roussopoulos, M. Seltzer, and M. Welsh, "Hourglass: An Infrastructure for Connecting Sensor Networks and Applications," Harvard University, 2004.
[22] S. Guo, Z. Zhong, and T. He, "FIND: Faulty Node Detection for Wireless Sensor Networks," in Proceedings in the 7th ACM Conference on Embedded Networked Sensor Systems, 2009, pp. 253-266.
[23] "FIPA: The Foundation for Intelligent Physical Agents," http://www.fipa.org/repository/standardspecs.html.
[24] B. Chen, H. H. Cheng, and J. Palen, "Mobile-C: A Mobile Agent Platform for Mobile C/C++ Code," Software - Practice & Experience, vol. 36, pp. 1711-1733, 2006.
[25] Y.-C. Chou, D. Ko, and H. H. Cheng, "An Embeddable Mobile Agent Platform Supporting Runtime Code Mobility, Interaction and Coordination of Mobile Agents and Host Systems," Information and Software Technology, vol. 52, pp. 185-196, 2010.
[26] B. Chen, D. D. Linz, and H. H. Cheng, "XML-Based Agent Communication, Migration and Computation in Mobile Agent Systems," Journal of Systems and Software, vol. 81, pp. 1364-1376, 2008.
[27] H. H. Cheng, "Ch: A C/C++ Interpreter for Script Computing," C/C++ User's Journal, vol. 24, pp. 6-12, 2006.
[28] "VIA ARTiGO A1100 PC Kit," http://www.via.com.tw/en/products/embedded/artigo/a1100/index.jsp.
[29] "FLAG Programmable System on Chip (PSoC) Development System," http://www.flag.com.tw/book/bookinfo.asp?bokno=E8773.



Embedded Systems - High Performance Systems, Applications and Projects

Edited by Dr. Kiyofumi Tanaka

ISBN 978-953-51-0350-9
Hard cover, 278 pages
Publisher InTech
Published online 16, March, 2012
Published in print edition March, 2012

Nowadays, embedded systems - computer systems that are embedded in various kinds of devices and play an important role of specific control functions, have permeated various scenes of industry. Therefore, we can hardly discuss our life or society from now onwards without referring to embedded systems. For wide-ranging embedded systems to continue their growth, a number of high-quality fundamental and applied researches are indispensable. This book contains 13 excellent chapters and addresses a wide spectrum of research topics of embedded systems, including parallel computing, communication architecture, application-specific systems, and embedded systems projects. Embedded systems can be made only after fusing miscellaneous technologies together. Various technologies condensed in this book as well as in the complementary book "Embedded Systems - Theory and Design Methodology", will be helpful to researchers and engineers around the world.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen