Mobile Agents for Ambient Intelligence

Ichiro Satoh

National Institute of Informatics,
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan
Tel: +81-3-4212-2546 Fax: +81-3-3556-1916
`ichiro@nii.ac.jp`

Abstract. This paper presents a mobile agent-based framework for ambient intelligence. The goal of the infrastructure is to provide people, places, and objects with computational functionalities to support and annotate them. Using location-tracking systems the infrastructure can navigate Java-based mobile agents to stationary or mobile computers near the entities and places to which the agents are attached, even when the locations change. The infrastructure enables application-specific functionalities to be implemented within mobile agents instead of the infrastructure itself. It maintains the locations of people and objects, including computing devices, and allows mobile users to directly access their personalized services from stationary computing devices in the environment or from their portable computing devices. This paper presents the rationale, design, implementation, and applications for our prototype infrastructure.

1 Introduction

Ambient intelligence enables us to be surrounded by electronic environments that are sensitive and responsive to people. Ambient intelligence technologies are expected to combine concepts of intelligent systems, perceptual technologies, and ubiquitous computing. Perceptual technologies have made it possible to measure and track the locations of people, computers, and practically any other object of interest. For example, indoor location systems, such as RFID (radio frequency identification) tags, detect the locations of physical entities in a building and enable applications to respond to these locations.

Location awareness is becoming an essential feature of services targeted at ambient intelligence. Several researchers have explored such location-aware services. Existing services can be classified into two approaches. The first concern is to make the computing devices that move with the user. It often assumes that such devices are attached to positioning systems, such as GPS receivers, which enable them to determine their own locations. For example, in HP's Cooltown project [5], mobile computing devices such as PDAs and smart phones are attached to GPSs to provide location-awareness for web-based applications running on the devices. The second approach assumes that a space is equipped with tracking systems which establish the location of physical entities, including people and objects, within the space so that application-specific services can be provided to appropriate computers. A typical example of this is the so-called follow-me application, which was a study by Cambridge University's Sentient Computing project

[3], to support ubiquitous and personalized services on computers located near users. The two approaches are posed as polar opposites, although their final goals seem to coincide.

This paper presents a mobile agent-based framework for integrating the two approaches to support their advantages. The framework does not distinguish between mobile and stationary computing devices. It can permit tracking sensors to be moved with the user and dynamically added to and removed from a space it does this, because it dynamically creates a world model when detecting the appearance and movement of sensors and physical entities including people, objects, and computing devices. Moreover, it is unique among other existing location-aware systems in that it uses mobile agent technology. It enables these agents to be spatially bound to people, places, and objects, which the agents then support and annotate. Using mobile agents makes the framework application-independent, application-specific services are implemented within mobile agents instead of the infrastructure. Therefore, the framework enables various location-aware or personalized services to be constructed.

In the remainder of this paper, we describe our design goals (Section 2), the design of our approach, called SpatialAgent, and a prototype infrastructure (Section 3), and programming models for mobile agents (Section 4). We describe the current implementation of the framework (Section 5) and present how to bridge the gap between the physical world and cyberspace (Section 6) and discuss our experience with several applications, which we developed with the infrastructure (Section 7). We briefly review related work (Section 8). We also provide a summary and some future issues (Section 9).

2 Approach

The framework presented in this paper aims to enhance the capabilities of users, particularly mobile users, things, including computing devices and non-electronic objects, and places, such as rooms, buildings and cities, that have the computational functionalities to support and annotate them.

2.1 Location-Sensing Systems

Our goal is to provide a location-aware system in which spatial regions can be determined within a few square feet, so that one or more portions of a room or building can be distinguished. The current implementation uses RFID tag technology to locate objects. An RFID system uses RF (radio frequency) readers, which detect the presence of small RF transmitters, often called *tags*. The framework assumes that physical entities, including people and computing devices, and places are equipped with these tags so that they are automatically locatable. It spatially binds software for information-based services to an RFID tag attached to a person, place, or thing in the physical world. The framework also provides a symbolic location model to hide the differences between the underlying location-sensing systems from applications as much as possible. This is because the framework aims at building location-aware applications for annotating and supporting people, objects, and places and such applications are often associated with semantic and structural spaces, such as buildings, rooms, and portions of a room or building, rather than geometric locations.

2.2 Location-Based Services

A ubiquitous computing environment consists of many computing devices, which may have only limited resources, such as restricted levels of CPU power and limited amounts of memory. As a result, even if a device is at a location suitable for providing a wanted service, the device may not be able to do so due to a lack of capabilities, such as input or output facilities. To overcome this limitation, the framework introduces mobile agent technology each agent only needs to be present at the computer during the time the computer needs the services provided by that agent. Mobile agent technology also has the following advantages.

- Various kinds of infrastructures have been used to construct and manage location-aware services. However, such infrastructures have mostly focused on a particular application, such as user navigation. By separating application-specific services from the infrastructure, our framework provides a general infrastructure for a variety of location-aware services, enabling application-specific services to be implemented within mobile agents.
- Each mobile agent can dynamically be deployed at and locally executed within computers near the position of the user. As a result, the agent can directly interact with the user, whereas RPC-based approaches, on which other approaches are often based, must have network latency between the local computer and remote servers. The agent also can directly access various equipment belonging to that device as long as the security mechanisms of the device permit this.
- After arriving at its destination, a mobile agent can continue working without losing any previous results of working, e.g., the content of instance variables in the agent's program, at the source computers. Thus, the technology enables us to easily build follow-me applications as proposed in [3].

The framework presented in this paper enables a physical entity and place to spatially bind with one or more mobile agent-based services. These services annotate and support the entities or places in the sense that the services can be dynamically deployed at stationary and mobile computing devices that are near or within the locations of the entities and places.

3 Design and Implementation

This framework provides the middleware infrastructure for managing location-sensing systems and deploying mobile agents at suitable computing devices according to the locations of users and objects, including computing devices. It consists of three parts: (1) location information servers, called LISs, (2) service-provider mobile agents, and (3) agent hosts, as we can see in Figure 1. The first part manages more than one location-sensor and recommends destinations to mobile agents. The second offers application-

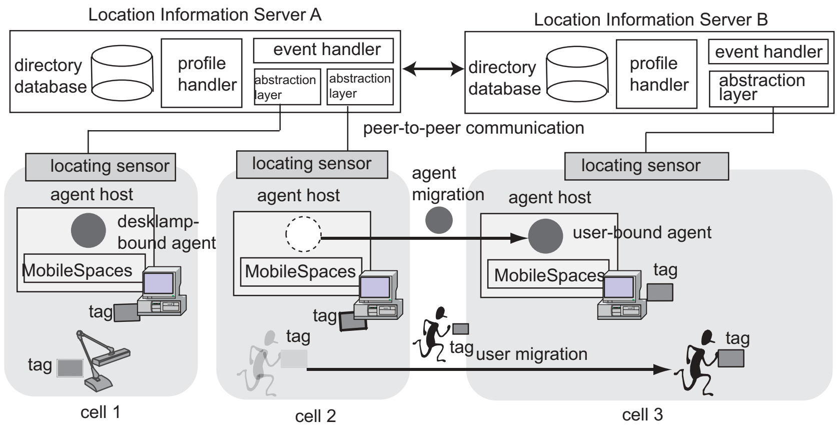

Fig. 1. Architecture of the SpatialAgent framework

specific services, which are attached to physical entities and places. The third consists of computing devices that can execute mobile agent-based applications.

3.1 Location Information Server

LISs are responsible for managing location sensing systems and recommending agents devices at which the agents provide their services. They can run on a stationary or mobile computer and provide all LISs that can run on a stationary or mobile computer and that have the following functionalities:

RFID-Based Location Model: This framework represents the locations of objects with a symbolic names to specifying the sensing ranges of RFID readers, instead of geographical models. Each LIS manages more than one RFID reader that detects the presence of tags and maintains up-to-date information on the identities of those that are within the zone of coverage. This is achieved by polling the readers or receiving events issued by the readers. An LIS does not require any knowledge on other LISs, but it needs to be able to exchange its information with others through multicast communication. To hide the differences between the underlying locating systems, each LIS maps low-level positional information from the other LISs into information in a symbolic model of location. An LIS represents an entity's location in symbolic terms of the RFID reader's unique identifier that detects the entity's tag. We call each RFID reader's coverage a *cell*, as in the models of location reported by several other researchers [7]. Multiple RFID readers in the framework do not have to be neatly distributed in spaces such as rooms or buildings to completely cover the spaces; instead, they can be placed near more than one agent host and the reader coverage can overlap.

Location Management: Each LIS is responsible for discovering mobile agents bound to tags within its cells. Each maintains a database in which it stores information about

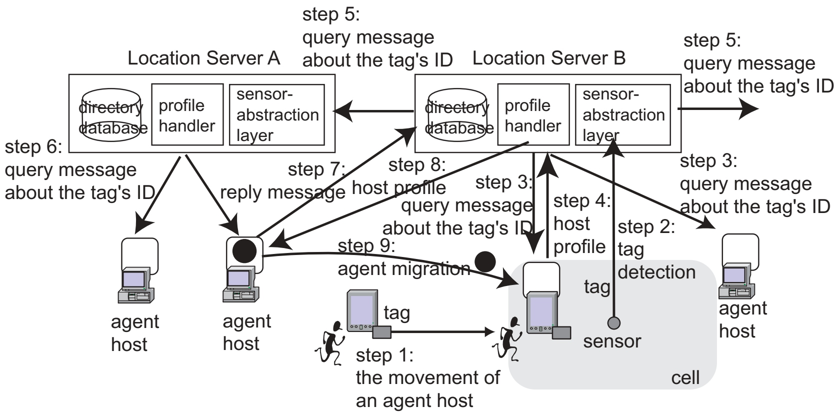

Fig. 2. Agent discovery and deployment

each of the agent hosts and each of the mobile agents attached to a tagged entity or place. When an LIS detects a new tag in a cell, the LIS multicasts a query that contains the identity of the new tag and its own network address to all the agent hosts in its current sub-network. It then waits for reply messages from the agent hosts. Here, there are two possible cases: the tag may be attached to an agent host or the tag may be attached to a person, place, or thing other than an agent host.

- In the first case, the newly arriving agent host will send its network address and device profile to the LIS; the profile describes the capabilities of the agent host, e.g., input devices and screen size. After receiving a reply message, the LIS stores the profile in its database and forwards the profile to all agent hosts within the cell.
- In the second case, agent hosts that have agents tied to the tag will send their network addresses and the requirements of acceptable agents to the LIS; requirements for each agent specify the capabilities of the agent hosts that the agent can visit and perform its services at.

The LIS then stores the requirements of the agents in its database and moves the agents to appropriate agent hosts in a manner we will discuss later. If the LIS does not have any reply messages from the agent hosts, it can multicast a query message to other LISs. When the absence of a tag is detected in a cell, each LIS multicasts a message with the identifier of the tag and the identifier of the cell to all agent hosts in its current sub-network. Figure 2 shows the sequence for migrating an agent to a suitable host when an LIS detects the presence of a new tag.

Location/Personal-Aware Deployment of Mobile Agents: We will now explain how the framework deploys agents at suitable agent hosts. When an LIS detects the movement of a tag attached to a person or thing to a cell, it searches its database for agent hosts that are present in the current cell of the tag. It also selects candidate destinations from the set of agent hosts within the cell, according to their respective capabilities. The framework offers a language based on CC/PP (composite capability/preference profiles) [16]. The language is used to describe the capabilities of agent hosts and the requirements of mobile agents in an XML notation. For example, a description contains information on the following properties of a computing device: vendor and model class of the device (i.e, PC, PDA, or phone), its screen size, the number of colors, CPU, memory, input devices, secondary storage, and the presence/absence of loudspeakers. The framework also allows each agent to specify the preferable capabilities of agent hosts that it may visit as well as the minimal capabilities in a CC/PP-based notation. Each LIS is able to determine whether or not the device profile of each agent host satisfies the requirements of an agent by symbolically matching and quantitatively comparing properties.

The LIS then unicasts a navigation message to each of the agents that are bound to the tagged entities or places, where the message specifies the profiles of those agent hosts that are present in the cell and satisfy the requirements of the agent. The agents are then able to autonomously migrate to the appropriate hosts. When there are multiple candidate destinations, each of the agents that is tied to a tag must select one destination based on the profiles of the destinations. When one or more cells geographically overlap, a tag may be in multiple cells at the same time and agents tied to that tag may then receive candidate destinations from multiple LISs. However, since the message includes the network address of the LIS, the agents can explicitly ask it about the cell ranges. Our goal is to provide physical entities and places with computational functionality from locations that are near them. Therefore, if there are no appropriate agent hosts in any of the cells at which a tag is present but there are some agent hosts in other cells, the current implementation of our framework forces agents tied to the tag to move to hosts in different cells.

3.2 Service-Provider Mobile Agent

The framework encapsulates application-specific services into mobile agents so that it is independent of any applications and can support multiple services. In the appendix of this paper, each mobile agent is constructed as a collection of Java objects and is equipped with the identifier of the tag to which it is attached.[1] Each is a self-contained program and is able to communicate with other agents. An agent that is attached to a user always internally maintains that user's personal information and carries all its internal information to other hosts. A mobile agent may also have one or more graphical user interfaces for interaction with its users. When such an agent moves to other hosts, it can easily adjust its windows to the new host's screen by using the compound document framework for the MobileSpaces system that was presented in our previous paper [10].

3.3 Agent Host

Each agent host must be equipped with a tag. It has two forms of functionality: one for advertising its capabilities and the other for executing and migrating mobile agents. The current implementation assumes that LISs and agent hosts can be directly connected through a wired LAN such as Ethernet or a wireless LAN such as IEEE802.11b. When a host receives a query message with the identifier of a newly arriving tag from an LIS,

[1] Appendix describes programming interfaces of agents.

it replies with one of the following three responses: (i) if the identifier in the message is identical to the identifier of the tag to which it is attached, it returns profile information on its capabilities to the LIS; (ii) if one of the agents running on it is tied to the tag, it returns its network address and the requirements of the agent; and (iii) if neither of the above cases applies, it ignores the message.

The current implementation of this framework is based on a Java-based mobile agent system called MobileSpaces [9].[2] Each MobileSpaces runtime system is built on the Java virtual machine, which conceals differences between the platform architecture of the source and destination hosts, such as the operating system and hardware. Each of the runtime systems moves agents to other agent hosts over a TCP/IP connection. The runtime system governs all the agents inside it and maintains the life-cycle state of each agent. When the life-cycle state of an agent changes, e.g., when it is created, terminates, or migrates to another host, the runtime system issues specific events to the agent. This is because the agent may have to acquire various resources or release them, such as files, windows, or sockets, which it had previously captured. When a notification on the presence or absence of a tag is received from a LIS, the runtime system dispatches specific events to the agents that are tied to that tag and these run inside it.

4 Service-Provider Mobile Agent Program

Services are implemented in mobile agents instead of the MobileSpaces runtime systems. Each mobile agent is executed in the MobileSpaces runtime system and migrated between the runtime systems running on different computers. It is constructed as a collection of Java objects and is equipped with the identifier of the tag to which it is attached. Every agent program must be an instance of a subclass of the abstract class `TaggedAgent` as follows:

```
class TaggedAgent extends MobileAgent
implements Serializable {
  void go(URL url) throws NoSuchHostException { ... }
  void duplicate() throws IllegalAccessException { ... }
  void destroy() { ... }
  void setTagIdentifier(TagIdentifier tid) { ... }
  void setAgentProfile(AgentProfile apf) { ... }
  URL getCurrentHost() { ... }
  boolean isConformableHost(HostProfile hpf) {...}
  ....
}
```

Here are some of the methods defined in the `TaggedAgent` class. An agent executes the `go(URL url)` method to move to the destination host specified as the `url` by its runtime system.[3] The `setTagIdentifier` method ties the agent to the identity of

[2] The framework itself is independent of the MobileSpaces mobile agent system and can thus work with other Java-based mobile agent systems.

[3] In MobileSpaces, agents can have higher-level routings among multiple hosts [12].

the tag specified as `tid`. Each agent can specify a requirement that its destination hosts must satisfy by invoking the `setAgentProfile()` method, with the requirement specified as `apf`. The class has a service method called `isConformableHost()`, which the agent uses to decide whether or not the capabilities of the agent hosts specified as an instance of the `HostProfile` class satisfy the requirements of the agent. Each agent can have more than one listener object that implements a specific listener interface to hook certain events issued before or after changes in its life-cycle state or the movements of its tag.

5 Current Status

The framework presented in this paper was implemented in Sun's Java Developer Kit, version 1.1 or later versions, including Personal Java. This section discusses some features of the current implementation.

5.1 Location-Sensing Systems

The current implementation supports four commercial RFID systems: RF Code's Spider system, Alien Technology's 915Mhz RFID-tag system, Philips' I-Code system, and Hitachi's mu-chip system. The first system provides active RF-tags, which periodically emit an RF-beacon that conveys their unique identifier (every second) via 305 MHz-radio pulse. The system allows us to explicitly control the omnidirectional range of each of the RF readers to read tags within a range of 1 to 20 meters. The Alien Technology system provides passive RFID-tags and its readers periodically scan for present tags within a range of 3 meters by sending a short 915 MHz-RF pulse and waiting for answers from the tags. The Philips and Hitachi RFID systems are passive RFID tag systems that can sense the presence of tags within a range of a few centimeters. Although there are many differences between the four, the framework abstracts these.

5.2 Performance Evaluation

Although the current implementation of the framework was not built for performance, we measured the cost of migrating a 3-Kbytes agent (zip-compressed) from a source host to the destination host recommended by the LIS. This experiment was conducted with two LISs and two agent hosts, each of which was running on one of four computers (Pentium III-1GHz with Windows 2000 and JDK 1.4), which were directly connected via an IEEE802.11b wireless network. The latency of an agent's migration to the destination after the LIS had detected the presence of the agent's tag was 410 msec and the cost of agent migration between two hosts over a TCP connection was 42 msec. The latency included the cost of the following processes: UDP-multicasting of the tags' identifiers from the LIS to the source host, TCP-transmission of the agent's requirements from the source host to the LIS, TCP-transmission of a candidate destination from the LIS to the source host, marshaling the agent, migrating the agent from the source host to the destination host, unmarshaling the agent, and security verification. We believe that this latency is acceptable for a location-aware system used in a room or building.

5.3 Security and Privacy

Security is essential in mobile agent computing. The framework can be built on many Java-based mobile agent systems with the Java virtual machine. Therefore, it can directly use the security mechanism of the underlying mobile agent system. The Java virtual machine can explicitly restrict agents so that they can only access specified resources to protect hosts from malicious agents. To protect against the passing of malicious agents between agent hosts, the MobileSpaces system supports a Kerberos-based authentication mechanism for agent migration. It authenticates users without exposing their passwords on the network and generates secret encryption keys that can selectively be shared between mutually suspicious parties.

The framework only maintains per-user profile information within those agents that are bound to the user. It promotes the movement of such agents to appropriate hosts near the user in response to his/her movement. Since agents carry their users' profile information within them, they must protect such private information while they are moving over a network.[4] The MobileSpaces system can transform agents into an encrypted form before migrating them over the network and decrypt them after they arrive at their destinations. Moreover, since each mobile agent is just a programmable entity, it can explicitly encrypt its particular inner fields and migrate itself with the fields along with its own cryptographic procedure, except for its secret keys.

6 Discussions

The framework does not have to distinguish between mobile and stationary computing devices and between mobile and stationary location-sensing systems. It can inform mobile agents attached to tags about their appropriate destinations according to the current positions of the tags. It supports four types of linkages between a physical entity such as a person, thing, or place, and one or more mobile agents, as shown in Fig. 3.

- The first type of linkage assumes that a moving entity carries more than one tagged agent host and that a space contains a place-bound tag and sensor (Fig. 3 a). When the sensor detects the presence of a tag that is bound to one of the agent hosts, the framework instructs the agents attached to the tagged place to migrate to the visiting agent hosts to offer location-dependent services.
- The second type of linkage assumes that tagged agent hosts and sensors have been allocated (Fig. 3 b). When a tagged moving entity enters the coverage area of one of the sensors, the framework instructs the agents attached to the entity to migrate to the agent hosts within the same coverage area to offer the entity-dependent services of the entity.
- The third type of linkage assumes that an entity carries a sensor and more than one agent host and that a space contains more than one place-bound tag (Fig. 3 c). When the entity moves near a place-bound tag and the sensor detects the presence of the

[4] The framework itself cannot protect agents from malicious hosts, because this problem is beyond the scope of this paper.

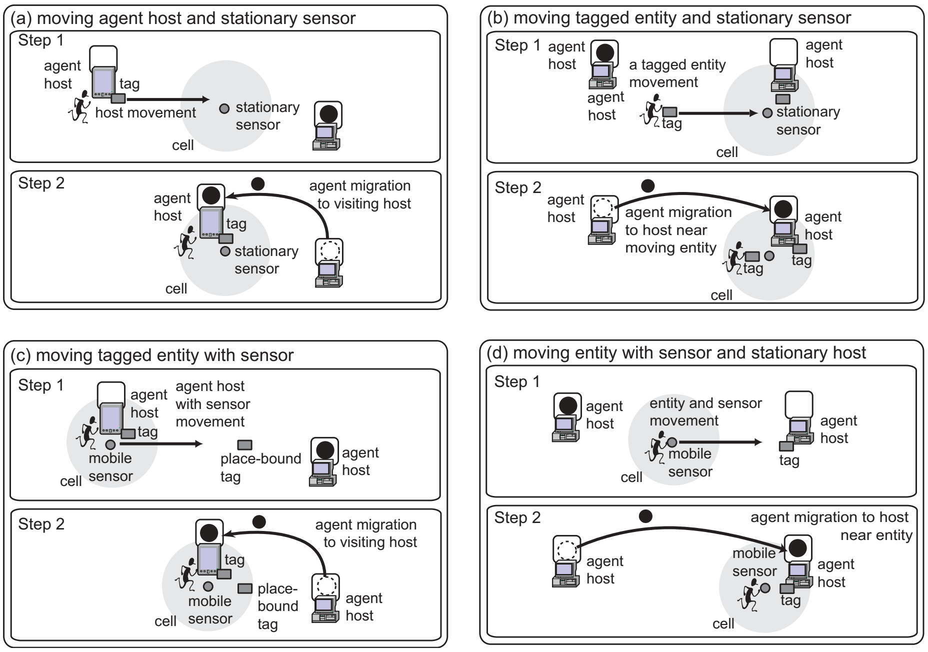

Fig. 3. Four types of linkages between physical and logical entities

tag within its coverage area, the framework instructs the agents attached to the tagged place to migrate to the visiting agent hosts to offer the location-dependent services of the place.

– The fourth type of linkage assumes that an entity carries a sensor and a space contains a place-bound tag and more than one tagged agent host (Fig. 3 d). When the entity moves and the reader detects the presence of an agent host's tag within its coverage area, the LIS instructs the agents attached to the moving entity to migrate to the agent host within the same coverage area to offer services dependent on the entity.

Note that the above linkages are independent of the underlying locating systems. Therefore, they are available in various source of location information, e.g., GPS, local wireless networks, and cellular networks. Existing location-aware systems can only support each of the above linkages, whereas our infrastructure does not have to distinguish between them and can synthesize them seamlessly. For example, the linkage shown in Fig. 3 (a) corresponds to *person tracking display* approach in the EasyLiving project [1], the linkage shown in Fig. 3 (b) corresponds to *Follow-me applications* approach in the Sentient Computing project [3] and the linkage shown in Fig. 3 (c) corresponds to services on location-aware portable devices studied in the Cooltown [5] and NEXUS [4] projects.

7 Applications

This section presents several typical location-based and personalized services that were developed through the framework. Note that these services can be executed at the same time, since the framework itself is independent of any application-specific services and each service is implemented within mobile agents.

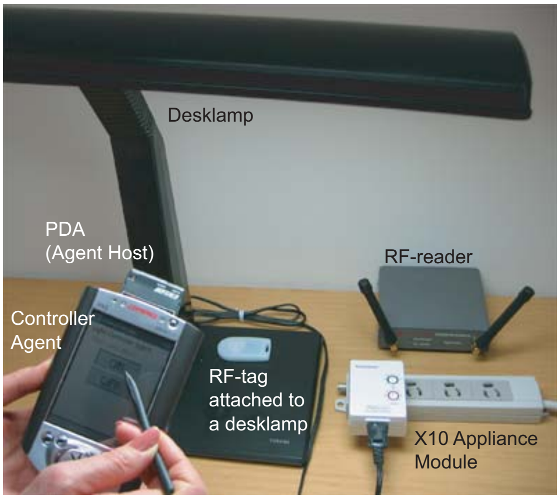

Fig. 4. Controlling desk lamp from PDA

7.1 Location-Bound Universal Remote Controller

The first example corresponds to Figure 3 (a) and allows us to use a PDA to remotely control nearby electric lights in a room. Each light was equipped with a tag and was within the range covered by an RFID reader in the room. We controlled power outlets for lights through a commercial protocol called X10. In both approaches described here, the lights were controlled by switching their power sources on or off according to the X10 protocol. In this system, place-bound controller agents, which can communicate with X10-base servers to switch lights on or off, are attached to locations with room lights. Each user has a tagged PDA, which supports the agent host with WindowsCE and wireless LAN interface. When a user with a PDA visits a cell that contains a light, the framework moves a controller agent to the agent host of the visiting PDA. The agent, now running on the PDA, displays a graphical user interface to control the light. When the user leaves that location, the agent automatically closes its user interface and returns to its home host.

7.2 Mobile Personal Assistance

The second example corresponds to Figure 3 (b) and offers a user assistant agent that follows its user and maintains profile information about him/her inside itself, so that

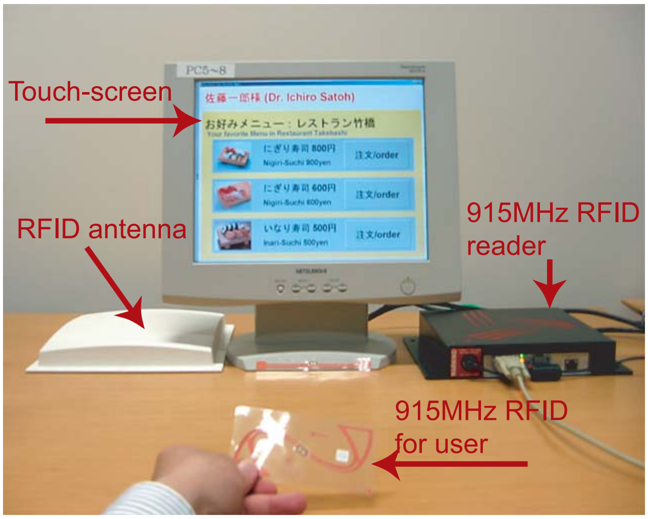

Fig. 5. Screenshot of follow-me user assistant agent for selecting its user's favorite sushi from the menu database of a restaurant that the user is in front of

he/she can always assist the agent in a personalized manner anywhere. Suppose that a user has a 915MHz-RFID tag and is moving on front of a restaurant, which offers an RFID reader and an agent host with a touch-screen. When the tagged user enters inside the coverage area of the reader, the framework enables his/her assistant agents to move to the agent host near his/her current location. After arriving at the host, the agent accesses a database provided in the restaurant to obtain a menu from the restaurant. [5] It then selects appropriate meal candidates from the menu according to his/her profile information, such as favorite foods and recent experiences, stored inside it. It next displays only the list of selected meals on the screen of its current agent host in a personalized manner for him/her. Figure 5 shows that a user's assistant agent runs on the agent host of the restaurant and seamlessly embeds a list of pictures, names, and prices of selected meal candidates with buttons for ordering them into its graphical user interface. Since a mobile agent is a program entity, we can easily define a more intelligent assistant agent.

7.3 User Navigation System

We developed a user navigation system that assists visitors to a building. Several researchers have reported on other similar systems [2, 4]. In our system, tags are distributed to several places within a building, such as its ceilings, floors, and walls. As we can see from Figure 3 (c), each visitor carries a wireless-LAN enabled tablet PC, which is equipped with an RFID reader to detect tags, and includes an LIS and an agent

[5] The current implementation of the database maintains some information about each available food, such as name and price, in an XML-based entry.

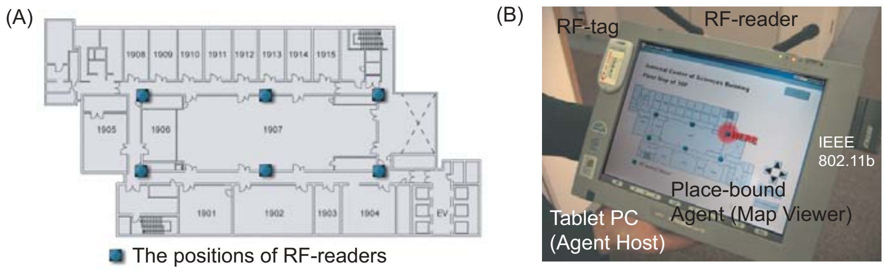

Fig. 6. A) Positions of RF-tags in floor and B) screen-shot of map-viewer agent running on table PC

host. The system initially deploys place-bound agents to invisible computers within the building. When a tagged position is located by a cell of the moving RFID reader, the LIS running on the visitor's tablet PC detects the presence of the tag. The LIS detects the place-bound agent that is tied to the tag. It then instructs the agent to migrate to its agent host and provide the agent's location-dependent services at the host. The system enables more than one agent tied to a place to move to the table PC. The agents then return to their home computers and other agents, which are tied to another place, may move to the tablet PC. Fig. 6 shows a place-bound agent to display a map of its surrounding area on the screen of a tablet PC.

8 Related Work

Research on ambient intelligence or smart spaces has become common at many universities and corporate research facilities. For example, Cambridge University's Sentient Computing project [3] provides a platform for building location-aware applications by using locating systems. Unlike our framework, the management of the the platform is centralized and cannot dynamically reconfigure itself. The project uses CORBA-based middleware can move CORBA objects to hosts according to the location of tagged objects [8]. However, CORBA objects are not well suited for implementing user interface components and migrating between heterogeneous platforms. Microsoft's EasyLiving project [1] provides context-aware spaces, with a particular focus on home and office. This project can dynamically aggregate network-enabled input/output devices, such as keyboards and mice, even when they belong to different computers in a space. However, its management is centralized, and it does not dynamically migrate software to other computers according to the location of users. Both projects assume that locating sensors have initially been allocated in the room, and that dynamically reconfiguring the platform is difficult when sensors are added to or removed from the environment. In contrasts, our framework permits sensors to be mobile and scattered throughout a space.

Several studies have focused on enhancing context-awareness in mobile computing, for example HP's Cooltown [5] and the NEXUS system [4]. These projects assume that each user has a notebook PC, tablet PC, or PDA, equipped with GPS-based positioning sensors and wireless communication. Applications that run on such devices access resources stored on the web via a browser by using standard HTTP communication.

Although user familiarity with web browsers is an advantage in these systems, the services available are constrained by the limitations of web browsers and HTTP. In contrast, our framework can dynamically deploy mobile agents, which are autonomous programmable entities, to mobile computing devices. Unlike our approach, neither Cooltown nor NEXUS can support mobile users through stationary computers distributed in a smart environment.

While many researchers have explored mobile agent technology, no attempts have been made to integrate the mobility of physical objects with the mobility of agents in a ubiquitous computing setting. We previously presented an early prototype of the present framework [11] that did not support the mobility of sensors and agent hosts, so that the three linkages described in the second section of this paper were not available in that previous version of the framework.

9 Conclusion

We presented a mobile agent-based infrastructure for managing location-sensing systems and dynamically deploying services at suitable computing devices. Using location-tracking systems the infrastructure provides entities, e.g. people and objects, and places, with mobile agents to support and annotate them and migrate agents to stationary or mobile computers near the locations of the entities and places to which the agents are attached. It is a general framework in the sense that it is independent of any higher-level applications and location-sensing systems and supports a variety of spatial linkages between the physical mobility of people and things and the logical mobility of services. Furthermore, we designed and implemented a prototype system of the infrastructure and demonstrated its effectiveness in several practical applications.

Finally, we would like to point out further issues to be resolved. Since the framework presented in this paper is general-purpose, in future work we need to apply it to specific applications as well as the three applications presented in this paper. The location model of the framework was designed for operating real location-sensing systems in ubiquitous computing environments. We plan to design a more elegant and flexible world model for representing the locations of people, things, and places in the real world by incorporating existing spatial database technologies. We have developed an approach to testing context-aware applications on mobile computers [13, 14]. We are interested in developing a methodology that would test applications based on the framework. We have also constructed a general location model for ambient intelligence [15]. We plan to integrate the model into the location-aware deployment of services presented in the paper.

References

1. B. L. Brumitt, B. Meyers, J. Krumm, A. Kern, S. Shafer: EasyLiving: Technologies for Intelligent Environments, Proceedings of International Symposium on Handheld and Ubiquitous Computing, pp. 12-27, 2000.

2. K. Cheverst, N. Davis, K. Mitchell, and A. Friday: Experiences of Developing and Deploying a Context-Aware Tourist Guide: The GUIDE Project, Proceedings of Conference on Mobile Computing and Networking (MOBICOM'2000), pp. 20-31, ACM Press, 2000.
3. A. Harter, A. Hopper, P. Steggeles, A. Ward, and P. Webster: The Anatomy of a Context-Aware Application, Proceedings of Conference on Mobile Computing and Networking (MOBICOM'99), pp. 59-68, ACM Press, 1999.
4. F. Hohl, U. Kubach, A. Leonhardi, K. Rothermel, and M. Schwehm: Next Century Challenges: Nexus - An Open Global Infrastructure for Spatial-Aware Applications, Proceedings of Conference on Mobile Computing and Networking (MOBICOM'99), pp. 249-255, ACM Press, 1999).
5. T. Kindberg, et al: People, Places, Things: Web Presence for the Real World, Technical Report HPL-2000-16, Internet and Mobile Systems Laboratory, HP Laboratories, 2000.
6. B. D. Lange and M. Oshima: Programming and Deploying Java Mobile Agents with Aglets, Addison-Wesley, 1998.
7. U. Leonhardt, and J. Magee: Towards a General Location Service for Mobile Environments, Proceedings of IEEE Workshop on Services in Distributed and Networked Environments, pp. 43-50, IEEE Computer Society, 1996.
8. D. Lopez de Ipina and S. Lo: LocALE: a Location-Aware Lifecycle Environment for Ubiquitous Computing, Proceedings of Conference on Information Networking (ICOIN-15), IEEE Computer Society, 2001.
9. I. Satoh: MobileSpaces: A Framework for Building Adaptive Distributed Applications Using a Hierarchical Mobile Agent System, Proceedings of Conference on Distributed Computing Systems (ICDCS'2000), pp. 161-168, IEEE Computer Society, 2000.
10. I. Satoh: MobiDoc: A Framework for Building Mobile Compound Documents from Hierarchical Mobile Agents, Proceedings of Symposium on Agent Systems and Applications / Symposium on Mobile Agents (ASA/MA'2000), LNCS, Vol. 1882, pp. 113-125, Springer, 2000.
11. I. Satoh: SpatialAgents: Integrating User Mobility and Program Mobility in Ubiquitous Computing Environments, Wireless Communications and Mobile Computing, vol.3, no.4, pp.411-423, Wiley, June 2003.
12. I. Satoh: Building Reusable Mobile Agents for Network Management, IEEE Transactions on Systems, Man and Cybernetics, vol.33, no. 3, part-C, pp.350-357, August 2003.
13. I. Satoh: A Testing Framework for Mobile Computing Software, IEEE Transactions on Software Engineering, vol. 29, no. 12, pp.1112-1121, December 2003.
14. I. Satoh: Software Testing for Wireless Mobile Computing, IEEE Wireless Communications, vol. 11, no. 5, pp.58-64, IEEE Communication Society, October 2004.
15. I. Satoh: Location Model for Pervasive Computing Environments, to appear in Proceedings of IEEE 3rd International Conference on Pervasive Computing and Communications (PerCom'05), IEEE Computer Society, March 2005.
16. World Wide Web Consortium (W3C): Composite Capability/Preference Profiles (CC/PP), http://www.w3.org/TR/NOTE-CCPP, 1999.