Location-Based Services in Ubiquitous Computing Environments

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 framework for providing dynamically deployable services in ubiquitous computing settings. The framework attaches physical entities and spaces with application-specific services to support and annotate them. By using RFID-based tracking systems, it detects the locations of physical entities, such as people or things, and deploys services bound to the entities at proper computing devices near the locations of the entities. It enables location-based services to be implemented as mobile agents and operated at stationary or mobile computing devices, which are at appropriate locations, even if the services do not have any location-information. The paper also describes a prototype implementation of the framework and several practical applications.

1 Introduction

As Mark Weiser envisioned [20], a goal of ubiquitous computing is to provide various services by making multiple computers available throughout the physical environment, but, in effect, making them invisible to the user. Another goal of ubiquitous computing is for it to integrate the physical world with cyberspace. Actually, perceptual technologies have made it possible to detect the presence or positions of people and any other object we care to think about. Context-awareness, in particular user-awareness and location-awareness, is becoming an essential feature of services that assist our everyday lives in ubiquitous computing environments.

However, ubiquitous computing devices are not suitable for providing multiple-purpose and personalized services, because most devices tend to have limited storage and processing capacity and are thus incapable of internally maintaining a variety of software and profile databases on the users. In fact, although there have been many attempts to develop location-based services thus far, most existing location-based systems have inherently focused on particular services, such as user navigation for visualizing locations on maps and information providing the information relevant to the user's current location. As a result, it is difficult for these systems to support other services for which they were not initially designed. Furthermore, they are often implemented in an ad-hoc manner with centralized management. Therefore, they cannot dynamically reconfigure themselves when new services are needed.

This paper presents a framework for deploying and operating location-based applications to solve these problems and this is based on two key ideas. The first is to introduce mobile agent technology as a mechanism to dynamically deploy services. Since many computing devices in ubiquitous computing environments only have limited resources, they cannot provide all services required due to limited computational resources, even if they are at suitable locations. Therefore, the framework provides an infrastructure for dynamically deploying service-provider agents to support services at computers that need the services. The second idea is to separate application-specific services from the infrastructure. Since each mobile agent is a programmable entity, the framework enables application-specific services, including user interfaces and application logic, to be implemented within mobile agents. Using mobile agents makes the framework independent of any applications, because application-specific services are implemented within mobile agents instead of the infrastructure. Since the infrastructure is responsible for automatically deploying mobile agents at appropriate computers, they can provide their services without any location-information.

In the remainder of this paper, we briefly review related work (Section 2), describe our design goals (Section 3), the design of our framework, called *SpatialAgent*, (Section 4), and an implementation of the framework (Section 5). We describe some experiences we had with several applications, which we used the framework to develop (Section 6). We provide a summary and discuss some future issues (Section 7).

2 Background

There have been many attempts to develop and operate location-based services. Existing services can be classified into two types of approaches.

The first is to make computing devices move with the user. It often assumes that such devices are attached to positioning systems, such as Global Positioning Systems (GPS) receivers. For example, HP's Cooltown project [7] is an infrastructure for bridging people, places, and things in the physical world with web resources that are used to store information about them. It allow users to access resources via browsers running on handheld computing devices. All the services available in the Cooltown system are constrained by limitations with web browsers and HTTP. Stuttgart University's NEXUS project [5] provides a platform that supports location-aware applications for mobile users with handheld devices, like the Cooltown project. Unlike our approach, however, both projects are not suitable for supporting mobile users from stationary computers distributed in a smart environment.

The second approach assumes that a space is equipped with tracking systems which establish the location of physical entities, including people and objects, within it so that application-specific services can be provided at appropriate computers. Cambridge University's Sentient Computing project [4] provides a location-aware platform using infrared-based or ultrasonic-based locating systems in a building. Using the VNC system [12], the platform can track the movement of a tagged entity, such as individuals and things, so that the graphical user interfaces of the user's applications follow the user while he/she is moving around. Since the applications must be executed in remote servers, the platform may have non-negligible interactive latency between the servers and hosts the user accesses locally. Recently, a CORBA-based middleware, called LocARE, has been proposed [10]. The middleware can move CORBA objects to hosts according to the location of tagged objects. Although the project provides similar functionality to that of our framework, its management is centralized and it is difficult to dynamically reconfigure the platform when sensors are added to or removed from the environment.

Microsoft's EasyLiving project [2] provides context-aware spaces, with a particular focus on the home and office. A computer-vision approach is used to track users within the spaces. Both the projects assume that locating sensors have initially been allocated in the room, and it is difficult to dynamically configure the platform when sensors are added to or removed from the environment, whereas our framework permits sensors to be mobile and scatteredly throughout the space.

ETH has developed an event-based architecture for managing RFID tags [13]. Like our framework, the architecture can link physical objects with software entities, called virtual counterparts. However, the goal of the architecture is to develop software frameworks that ease the development of particular applications rather than a general framework for supporting various applications. Although a ubiquitous computing environment is a distributed system whose computing devices may be dynamically added to and removed from the system, the architecture is managed by a centralized server, whereas our framework is essentially managed in a plug-and-play manner. Moreover, since the architecture cannot migrate its software entities among ubiquitous computing devices, it cannot effectively support moving objects in the physical world, unlike our framework.

We presented an early prototype of the present framework in a previous paper [17] and this was just an infrastructure for allowing Java-based agents to follow moving users through locating systems and did not encapsulate application-specific tasks into mobile agents, unlike the framework presented in this paper. Also, since the previous system did not support sensor mobility, it could not support all spatial linkages (Figure 1), whereas the framework presented in this paper was designed to be based on RFID-based sensors and it therefore permitted the mobility of sensors as well as physical entities, such as people, objects, and computing devices. We will also present an extension of the framework with the ability of managing various location sensors other than RFID-based sensors in an upcoming paper [18].

3 Approach

The goal of the framework presented in this paper is to provide a general infrastructure for supporting multiple location-aware and personalized services in ubiquitous computing environments.

3.1 Dynamically Deployable Services

Various kinds of infrastructures have been used to construct and manage location-aware services. However, such infrastructures have mostly focused either on a particular application or on a specific sensor technology. By separating application-specific services from infrastructures, our framework provides a general infrastructure for location-aware services and enables application-specific services to be implemented as mobile agents.

Each mobile agent can travel from computer to computer under its own control. When a mobile agent moves to another computer, not only the code but also the state of the agent is transferred to the destination. After arriving at a computer, agents can still continue their processes and access the resource provided by the computer as long as long as the security mechanisms of the computer permit this. Mobile agent technology also has the following advantages in ubiquitous and mobile computing settings:

– 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, where RPC-based approaches, which other existing approaches are often based on, must have network latency between computers and remote servers. It also can directly access various equipment, which belong to that device as long as security mechanisms permit this.
– After arriving at its destination, a mobile agent can continue working without losing the 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 [4].
– Mobile agents can help to conserve the limited resources of computing devices, since each agent only needs to be present at the devices while they are required to offer the services provided by that agent. Mobile agents also have the potential to mask disconnections in some cases. Once a mobile agent is completely transferred to a new location, it can continue its execution at the new location, even when the new location is disconnected from the source location.

3.2 Location Sensing Systems

This framework offers a location-aware system in which spatial regions can be determined within a few square feet, that distinguishes between one or more portions of a room or building. Existing location-based services are typically tailored to a particular type of tracking or positioning technology, such as GPS. The current implementation of the framework uses RFID technology as an alternate approach to locate objects. This is because RFID technologies are expected to be widely used in product distribution and inventory management and tags will placed on many low-cost items, including cans and books in the near future. An RFID system consists of RF (radio frequency) readers (so-called sensors or receivers), which detect the presence of small RF transmitters, often called *tags*. Advances in wireless technology enable passive RFID tags to be scanned over a few meters. For example, the Auto-ID center [1] and its sponsors are working to develop flexible tags and readers operated at ultra-high frequency (915 MHz in the US and 868 MHz in the EU). It expects that RFID tags will cost around 5 cents when produced in bulk and RFID readers around a hundred dollar in volume. The framework assumes that physical entities, including people, computing devices, and places will be equipped with RFID tags so that they are entities that are automatically locatable.

3.3 Architecture

The framework consists of three parts: (1) location information servers, called LISs, (2) mobile agents, and (3) agent hosts. The first provides a layer of indirection between the underlying RFID locating sensing systems and mobile agents. Each LIS manages more than one RFID reader and provides the agents with up-to-date information on the identifiers of RFID tags, which are present in the specific places its readers cover instead of on tags in the whole space. The second offers application-specific services, which are attached to physical entities and places, as collections of mobile agents. The third is a computing device that can execute mobile agent-based applications and issue specific events to the agents running in it when RFID readers detect the movement of the physical entities and places that the agents are bound to.

When an LIS detects a moving tag, it notifies mobile agents attached to it about the network addresses and capabilities of the candidate hosts that are near its location. Each of these agents selects one host from the candidate agent hosts recommended by the LIS and migrate to the selected host. The capabilities of a candidate host do not always satisfy all the requirements of an agent. Each agent does not need to have to know any information about the network addresses and locations of devices, which it may migrate to. This framework assumes that each agent itself should decide, on the basis of to its own configuration policy, whether or not it will migrate itself to the destination and adapt itself to the destination's capabilities.

Our final goal is widespread building-wide and city-wide deployment. It is almost impossible to deploy and administer a system in a scalable way when all of the control and management functions are centralized. LISs are individually connected to other servers in a peer-to-peer manner and exchange information with one another. LISs and agent hosts may be mobile and frequently shut down. The framework permits each LIS to run independently of the other LISs and it offers an automatic mechanism to register agent hosts and RFID readers. The mechanism requires agent hosts to be equipped with tags so that they are locatable.

3.4 Narrowing the Gap between Physical and Logical Mobilities

This framework can inform mobile agents attached to tags about their appropriate destinations according to the current positions of the tags. It supports three types of linkages between a physical entity such as a person, thing, or place, and one or more mobile agents as we can see in Figure 1.

– In the first linkage, a moving entity carries more than one tagged agent host and a space contains a place-bound RFID tag and readers. When the RFID reader detects the presence of a tag that is bound to one of the agent hosts, the framework instructs the agents that are attached to the tagged place to migrate to the visiting agent hosts to offer the location-based services the place has as we can see in Figure 1 (a).
– In the second linkage, tagged agent hosts and RFID readers are allocated. When a tagged moving entity enters the coverage area of one of the readers, the framework instructs the agents that are attached to the entity to migrate to the agent hosts within the same coverage area to offer the entity-dependent services the entity has as we can see in Figure 1 (b).
– In the third linkage, an entity carries an RFID reader and more than one agent host and a space contains more than one place-bound tag. When the entity moves to a nearby place-bound tag and the reader detects the presence of the tag within its

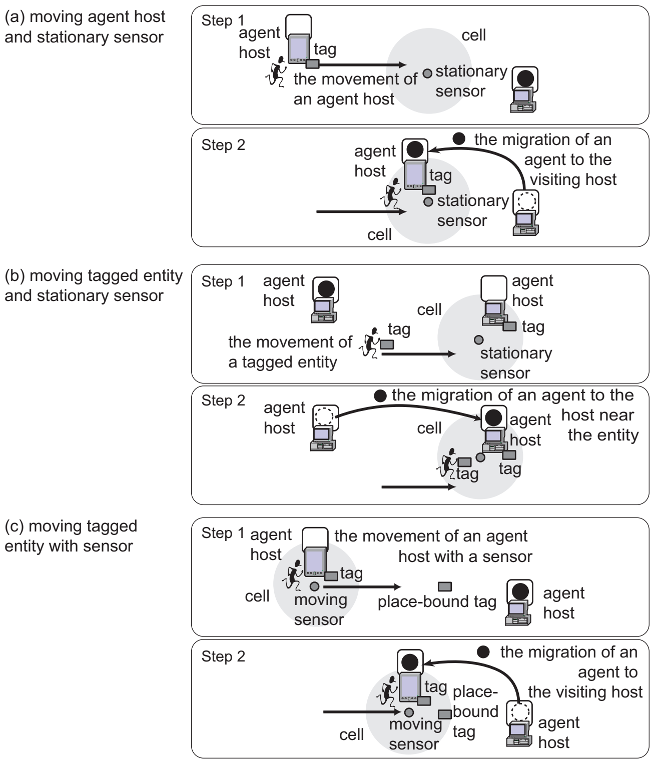
(a) moving agent host and stationary sensor
Step 1
agent host
tag
cell
the movement of an agent host
stationary sensor
Step 2
the migration of an agent to the visiting host
(b) moving tagged entity and stationary sensor
the movement of a tagged entity
the migration of an agent to the host near the entity
(c) moving tagged entity with sensor
the movement of an agent host with a sensor
moving sensor
place-bound tag

Fig. 1. Three linkages between physical and logical entities

coverage area, the framework instructs the agents that are attached to the tagged place to migrate to the visiting agent hosts to offer the location-dependent services the place has, as shown as we can see in Figure 1 (c).

Note that our framework does not have to distinguish between mobile and stationary computing devices and between mobile and stationary location-sensing systems.

4 Design

This section presents the design for the SpatialAgent framework and describes a prototype implementation of the framework. Figure 2 outlines the basic structure of the framework.

4.1 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:

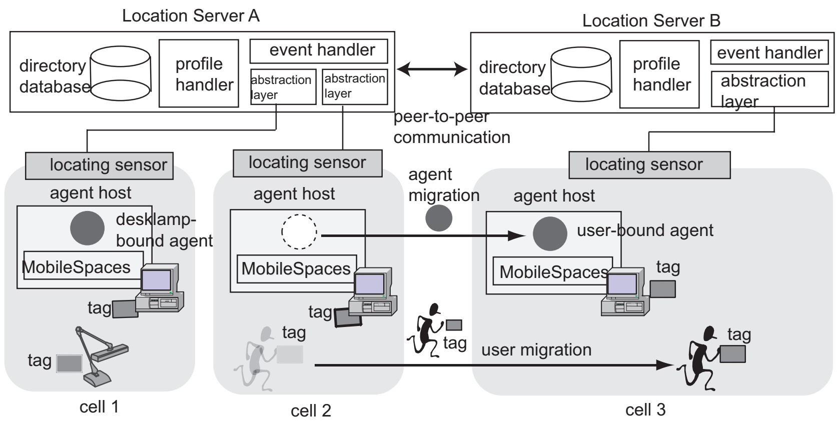

Fig. 2. Architecture of SpatialAgent Framework

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 [9]. 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 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 3 shows the sequence for migrating an agent to a suitable host when an LIS detects the presence of a new tag.

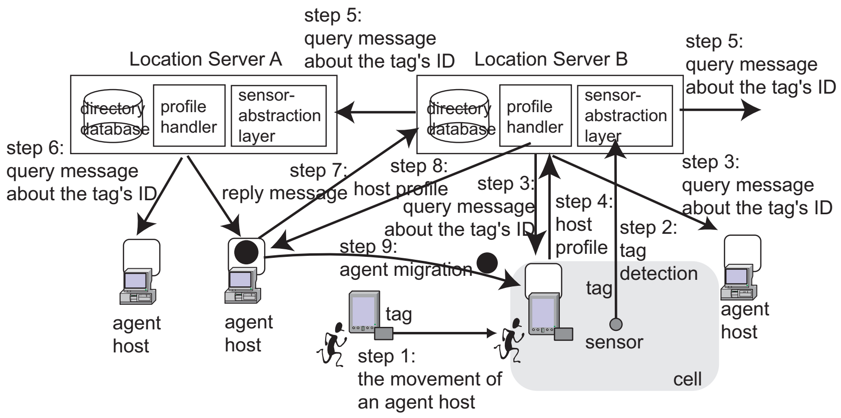

Fig. 3. Agent discovery and deployment

Spatial-dependent deployment of 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) [21]. 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.

4.2 Mobile Agent-Based Service-Provider

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 [15].

4.3 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, 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 [14].[2] Each MobileSpaces runtime system is built on the

[1] Appendix describes programming interfaces of agents.

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

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.

5 Implementation

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 Management of Locating 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.[3] 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 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.

6.1 Location-Bound Universal Remote Controller

The first example corresponds to Figure 1 (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

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

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.

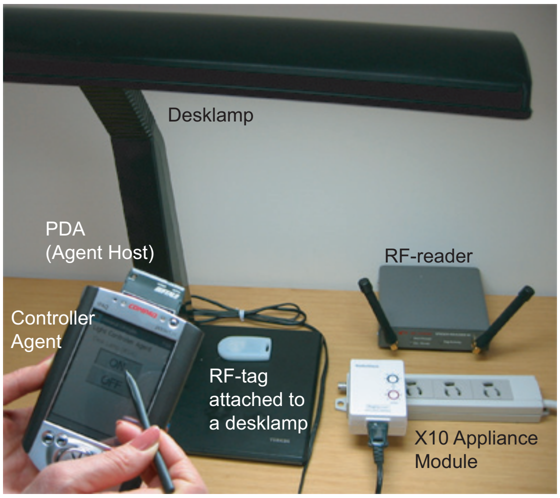

Fig. 4. Controlling desk lamp from PDA

6.2 Mobile Personal Assistance

The second example corresponds to Figure 1 (b) and offers a user assistant agent that follows its user and maintains profile information about him/her inside itself, so that 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. [4] 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.

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

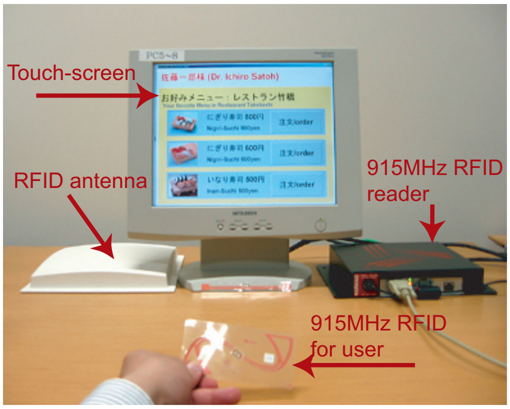

Fig. 5. Screenshot of follow-me user assistant agent for selecting its user's favorite sushi from the menu database of a restaurant that the user is in front of

6.3 User Navigation System

We developed a user navigation system that assists visitors to a building. Several researchers have reported on other similar systems [3,5]. 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 1 (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 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. Figure 6 shows a place-bound agent to display a map of its surrounding area on the screen of a tablet PC.

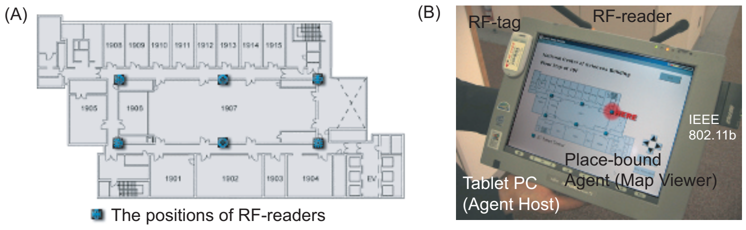

Fig. 6. (A) Positions of RF-tags in floor (B) and screen-shot of map-viewer agent running on table PC

7 Conclusion

We presented a framework for the development and management of location-aware applications in mobile and ubiquitous computing environments. The framework provides people, places, and things with mobile agents to support and annotate them. Using location-tracking systems, the framework can migrate mobile agents to stationary or mobile computers near the locations of the people, places, and things to which the agents are attached. The framework is decentralized and it is a generic platform independent of any higher-level applications and locating systems and supports stationary and mobile computing devices in a unified manner. We also designed and implemented a prototype system of the framework and demonstrated its effectiveness in several practical applications.

However, there are further issues that need to be resolved. Since the framework presented is general-purpose, we would need to apply it to specific applications in future work, 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 that represents the locations of people, things, and places in the real world by incorporating existing spatial database technologies.

References

1. Auto-ID center: http://www.autoidcenter.org/main.asp
2. 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.
3. 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.
4. 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.
5. 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).
6. K. Kangas and J. Roning: Using Code Mobility to Create Ubiquitous and Active Augmented Reality in Mobile Computing, Proceedings of Conference on Mobile Computing and Networking (MOBICOM'99), pp. 48–58, ACM Press, 1999.
7. 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.
8. B. D. Lange and M. Oshima: Programming and Deploying Java Mobile Agents with Aglets, Addison-Wesley, 1998.
9. 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.

10. 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.
11. N. Minar, M. Gray, O. Roup, R. Krikorian, and P. Maes: Hive: Distributed agents for networking things, Proceedings of Symposium on Agent Systems and Applications / Symposium on Mobile Agents (ASA/MA'99), IEEE Computer Society, 2000.
12. T. Richardson, Q, Stafford-Fraser, K. Wood, A. Hopper: Virtual Network Computing, IEEE Internet Computing, Vol. 2, No. 1, 1998.
13. K. Romer, T. Schoch, F. Mattern, and T. Dubendorfer: Smart Identification Frameworks for Ubiquitous Computing Applications, IEEE International Conference on Pervasive Computing and Communications (PerCom'03), pp.253–262, IEEE Computer Society, March 2003.
14. I. Satoh: MobileSpaces: A Framework for Building Adaptive Distributed Applications Using a Hierarchical Mobile Agent System, Proceedings of International Conference on Distributed Computing Systems (ICDCS'2000), pp. 161–168, IEEE Computer Society, 2000.
15. 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, September 2000.
16. I. Satoh: Flying Emulator: Rapid Building and Testing of Networked Applications for Mobile Computers, Proceedings of International Conference on Mobile Agents (MA'01), LNCS, Vol. 2240, pp. 103–118, Springer, December 2001.
17. I. Satoh: Physical Mobility and Logical Mobility in Ubiquitous Computing Environments, Proceedings of International Conference on Mobile Agents (MA'02), LNCS, Vol. 2535, pp. 186–202, Springer, October 2002.
18. I. Satoh: Linking Physical Worlds to Logical Worlds with Mobile Agents, Proceedings of International Conference on Mobile Data Management (MDM 2004), IEEE Computer Society, January 2004.
19. R. Want, A. Hopper, A. Falcao, and J. Gibbons: The Active Badge Location System, ACM Transactions on Information Systems, vol.10, no.1, pp. 91–102 ACM Press, 1992.
20. M. Weiser: The Computer for the 21st Century, Scientific American, pp. 94–104, September, 1991.
21. World Wide Web Consortium (W3C): Composite Capability/Preference Profiles (CC/PP), http://www.w3.org/TR/NOTE-CCPP, 1999.

Appendix: Service Provider Programs

This section explains the programming interface for service providers, which are implemented as mobile agents. Every agent program must be an instance of a subclass of the abstract class `TaggedAgent` as follows:

```
class TaggedAgent extends Agent 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 hfs) { ... }
  CellProfile getCellProfile(CellIdentifier cid)
    throws NoSuchCellException { ... }
  ....
}
```

Let us explain 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. The `duplicate()` method creates a copy of the agent, including its code and instance variables. The `setTagIdentifier` method ties the agent to the identity of 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 named `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. Also, the `getCellProfile()` method allows an agent to investigate the measurable range and types of RFID readers specified as `cid`.[5]

Each agent can subscribe to the types of events they are interested in and have more than one listener object that implements a specific listener interface to hook certain events. The following program is the definition of a lister object for receiving events issued before or after changes in its life-cycle state or movements of its tag.

```
interface TaggedAgentListener extends AgentEventListener {
  // invoked after creation at url
  void agentCreated(URL url);
  // invoked before termination
  void agentDestroying();
  // invoked before migrating to dst
  void agentDispatching(URL dst);
  // invoked after arrived at dst
  void agentArrived(URL dst);
  // invoked after the tag arrived at another cell
  void tagArrived(HostProfile[] apfs, CellIdentifier cid);
  // invoked after the tag left rom the current cell
  void tagLeft(CellIdentifier cid);
  // invoked after an agent host arrived at the current cell
  void hostArrived(AgentProfile apfs, CellIdentifier cid);
  ....
}
```

The above interface specifies the fundamental methods that are invoked by the runtime system when agents are created, destroyed, or migrate to another agent host. If a tagged entity or place is detected for the first time, the agent associated with that object or place has to be instantiated and then its `agentCreated()` method is invoked. Also, the `tagArrived()` callback method is invoked after the tag to which the agent is bound has entered another cell, to obtain the device profiles of agent hosts that are present in the new cell. The `tagLeft()` method is invoked after the tag is no longer in a cell for a specified period of time. The `agentDispatching()` method is invoked before the agent migrates to another host and the `agentArrived()` method is invoked after the agent arrives at the destination.

[5] The identifier of each RFID reader can be represented in a string format so that the framework can easily manage various RFID systems even when the identifiers of readers in these systems are different.