# Physical Mobility and Logical Mobility in Ubiquitous Computing Environments

Ichiro Satoh

National Institute of Informatics /
Japan Science and Technology Corporation
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 building context-aware applications in ubiquitous and mobile computing settings. The framework provides people, places, and things with computational functionalities to support and annotate them. It is unique among existing systems because the functionalities are implemented by mobile agents. Using location-tracking systems, this framework can navigate mobile agents to stationary or mobile computers near the locations of the entities and places to which the agents are attached, even when the locations change. The framework provides a way for mobile agents to follow their users as they move about and to adhere to places as virtual Post-its. A prototype implementation of the framework has been built on a Java-based mobile agent system and tested with several practical applications, including follow-me applications and a user- navigation system.

## 1 Introduction

Ubiquitous computing and mobile computing will be key areas in future computing. However, the two approaches have their own advantages and disadvantages. The concept of ubiquitous computing implies computation with elements that are contained in the environment rather than carried on the person. Various computing and sensing devices are in fact already present in almost every room of a modern building or house and in many of the public facilities of cities. They may now be disappearing inside all sorts of appliances and thus integrate with every aspect of life. This demonstrates the suitability of ubiquitous computing to provide environmental information and services. However, this approach is not suited to providing multiple-purpose and personalized services, because the devices embedded in various items within the environment tend to have limited storage and processing capacities. They are thus incapable of internally maintaining a variety of software and profile databases on the potential users. This approach may also raise serious privacy issues, because a ubiquitous computing environment would be able to monitor the preferences and locations of individuals.

On the other hand, the concept of mobile computing can mean that computing devices, for example, notebook-PCs, PDAs, wearable computers, are carried by users rather than contained within the environment. Recently, portable computing devices have become very small and powerful, giving their users access to a variety of applications in personalized form, regardless of the user locations. Each of these devices is intended to stay with a particular user so the user's profile can be maintained in the portable device and can it easily evolve over time, without having to be transferred from place to place in an external environment. Therefore, the mobile computing approach provides both personalization and privacy. However, its users are forced to carry devices, such as PCs, PDAs, and smart-phones, which may not be light and may only have small screens and clamped keyboards. Moreover, this approach is not suitable for context-dependent services because it is difficult for a portable device to sense its environment.

The two approaches are posed as polar opposites. We have attempted to alleviate the disadvantages of each approach by using the advantages of the other. Therefore, this paper presents a location-aware framework, called SpatialAgent, in which mobile agent technology is applied to provide a bridge between the two approaches. This framework enables mobile agents to be spatially bound to people, places, and things, which the agents support and annotate. Location-tracking systems are used within the framework to migrate such agents to stationary and mobile computing devices that are near the locations of the entities and places to which the agents are attached, even when the locations of the entities change.

Several ways of reducing the number of disadvantages of in both approaches have been explored. AT&T's Sentient Computing [3], for example, proposed a so-called follow-me application to support the provision of personalized services in ubiquitous computing settings. With HP's Cooltown [6], mobile computing devices such as PDAs and smart phones are attached to positioning sensors to provide location-awareness to web-based applications running on the devices. In contrast to these approaches, the framework presented in this paper does not distinguish between mobile and ubiquitous computing. Since mobile agents can travel between computers, the framework can naturally map the movements of physical entities such as people and objects to the movements of mobile agents in mobile and ubiquitous computing systems.

In the remainder of this paper, we describe our design goals (Section 2), the design of our framework, called SpatialAgent, and a prototype implementation of the framework (Section 3). We also discuss our experience with several applications that we developed by using the framework (Section 4), and briefly review related work (Section 5). We briefly discuss some future issues (Section 6) and provide a summary (Section 7).

## 2 Approach

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

### 2.1 Locating Systems

Our goal is to offer a location-aware system in which spatial regions can be determined to within a few square feet, so that one or more portions of a room or building can be distinguished. The framework itself is designed to be independent of any particular locational infrastructure and is accompanied by more than one locating system. It determines the positions of objects by identifying the spatial regions that contain the objects. In general, such locating systems consist of RF (radio frequency) or infrared sensors, which detect the presence of small RF or infrared transmitters, often called *tags*, each of which periodically transmits a unique identifier. The framework assumes that physical entities and places are equipped with their own unique tags so that they are automatically locatable entities.

The framework consists of two parts: (1) mobile agents and (2) location information servers, called LISs. The former offers application-specific services, which are attached to physical entities and places, as collections of mobile agents. The latter provide a layer of indirection between the underlying locating sensing systems and mobile agents. Each LIS manages more than one sensor and provides the agents with up-to-date information on the state of the real world, such as the locations of people, places, and things, and the destinations that the agents should migrate to.

### 2.2 Application-Specific Services

This framework enables application-specific services to be implemented as mobile agents. Mobile agent technology also has the following advantages in ubiquitous and mobile computing settings.

– After arriving at its destination, a mobile agent can continue working without losing the work results, for example the content for instance variables in the agent's program, at the source computers. Thus, the technology enables us to easily build follow-me applications as proposed by Cambridge University [3].
– Mobile and ubiquitous computers often have only limited resources, such as fixed levels of CPU power and restricted memory. Mobile agents can help to conserve these limited resources, since each mobile agent needs to be present at a computer only when the computer needs the services provided by that agent.
– Each mobile agent is locally executed on the computing device it is visiting and is able to directly access various equipment, which belong to that device as long as the security mechanisms of the device permits this.

In this framework, each mobile agent can be tied to radio-ID or infrared-ID tag attached to a person, place, or thing in the physical world.

### 2.3 Narrowing the Gap between Physical and Logical Mobility

This framework can inform mobile agents attached to tags about their proper destinations according to the current position of the tags. We call computing devices that can execute mobile agent-based applications *agent hosts*. This framework permits agent hosts to be mobile or stationary, but each host needs to be equipped with its own tag and must advertise its profile information to the LISs that detect the tag. The framework supports two types of linkages between a physical entity or place and more than one mobile agent:

– The framework binds one or more mobile agents to a tag, which is attached to a moving entity such as a user and a non-electronic object. When a tagged entity moves within a place, the framework prompts agents, which are bound to the moving entity, to move to appropriate stationary hosts within the same place, as shown in Fig. 1.

– The framework allows physical places to have their own agents which support location-dependent services. When a user with network-enabled computing devices is in a given place, the framework instructs the agents that are attached to the place to migrate themselves to the visiting devices, where they provide the location-dependent services of the place as shown in Fig. 2.

This framework permits a combination of both forms of linkages, while existing related work, such as AT&T's Sentient Computing and HP's Cooltown, only support one of them. In addition to this, the framework does not distinguish between mobile and stationary devices. In the framework, multiple sensors do not have to be neatly distributed in a space such as rooms or buildings to completely cover the spaces; instead, they can be placed near more than one agent host and the coverage of sensors can overlap.

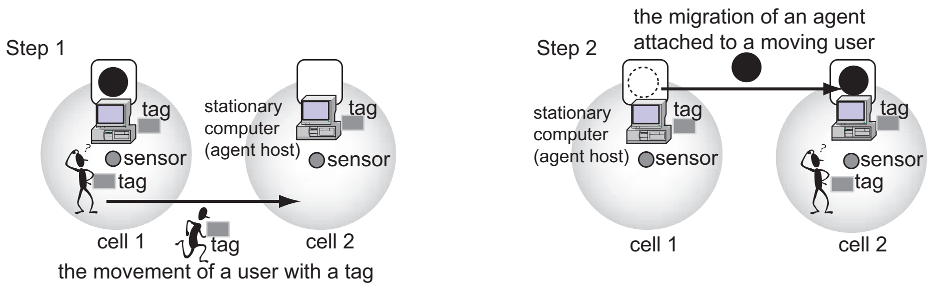
Step 1
tag
stationary computer (agent host)
tag
sensor
tag
sensor
cell 1
tag
cell 2
the movement of a user with a tag
Step 2
the migration of an agent attached to a moving user
stationary computer (agent host)
tag
sensor
tag
sensor
tag
cell 1
cell 2

**Fig. 1.** Migration of an agent, which is attached to a moving entity, to a computer at the current location of the entity

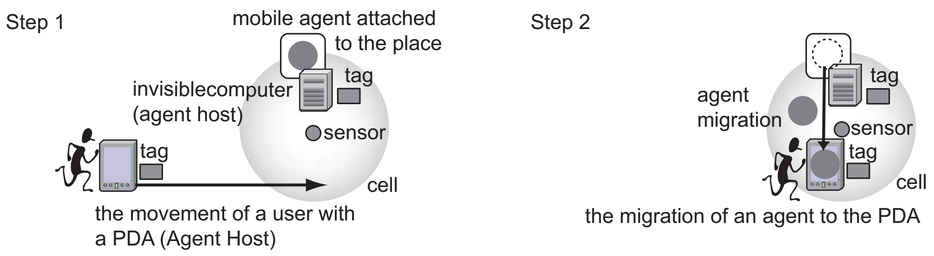
Step 1
mobile agent attached to the place
tag
invisiblecomputer (agent host)
sensor
tag
cell
the movement of a user with a PDA (Agent Host)
Step 2
tag
agent migration
sensor
tag
cell
the migration of an agent to the PDA

**Fig. 2.** Migration of an agent which is attached to a particular place to a computer visiting that place

## 2.4 Design Principles

In addition to achieving the goals presented above, the framework has the following advantages:

**Autonomy:** When an LIS detects the movement of a tag in the physical world, it informs agents bound to the tag about the network address and the capabilities of more than one candidate destination that the agents should visit, but the LIS itself does not send agents to a destination. Each of these agents selects one host among the candidate destinations recommended by the LIS and migrates to the selected host, since it is an autonomous entity. Moreover, when the capabilities of a candidate destination do not satisfy all the requirements of an agent, the agent itself should decide, on the basis of to its own configuration policy, whether or not it will migrate to the destination and adapt itself to the destination's capabilities.

**Scalability:** 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. Our framework consists of multiple servers, which are connected to individual servers in a peer-to-peer manner. Each LIS only maintains up-to-date information on the identifiers of tags, which are present in one or more of the specific places it manages, instead of on tags in the whole space.

**Extensibility:** LISs and agent hosts may be dynamically deployed and frequently shut down. The framework permits each LIS to run independently of the other LISs and offers an automatic mechanism for the registration of agent hosts. The mechanism requires agent hosts to be equipped with tags so that they are locatable and can advertise their capabilities.

**Reconfigurability:** In the framework, not only portable components but also system components, such as the sensors and agent hosts, are movable. As a result, it is almost impossible to maintain a geographical model of the whole system. To solve this problem, the framework provides a demand-driven mechanism for discovering the agents and agent hosts that are required, where the mechanism was inspired by ad-hoc mobile networking technology [12].

**Modularity and Application-Independence:** The framework should be as independent as possible of the underlying sensor technologies and mobile agent systems. This minimizes the effects of the distribution and heterogeneity of the underlying locating infrastructure on the applications. The framework itself is independent of application-specific tasks because such tasks are performed within mobile agents.

**Personalization and Privacy:** 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 the user's movement. Thus, the agents do not leak profile information on their users to other parties and can interact with their mobile users in personalized forms that have been adapted to respective individual users.

## 3 Design and Implementation

This section presents the design of the SpatialAgent framework and describes a prototype implementation of the framework. Fig. 3 shows the basic structure of the framework.

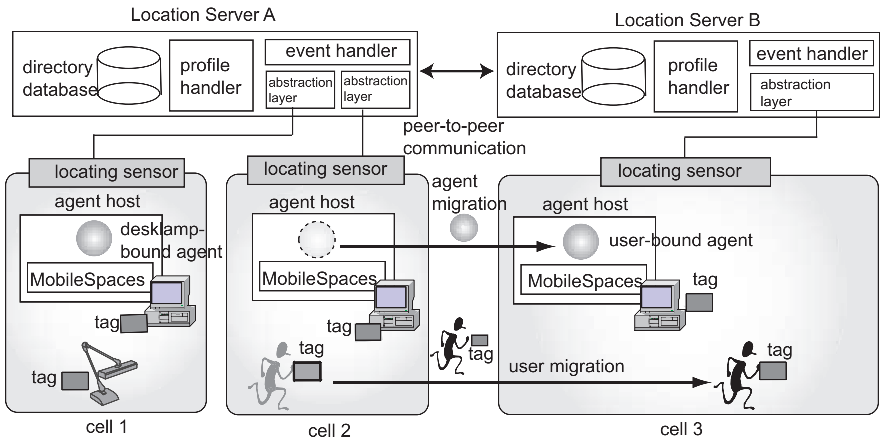

**Fig. 3.** Architecture of the SpatialAgent Framework

### 3.1 Location Information Server

Each LIS can run on a stationary or mobile computer and provides the following functionality:

**Management of Locating Sensors:** Each LIS manages multiple sensors that detect the presence of tags and maintains up-to-date information on the identities of tags that are within the zone of coverage by its sensors. This is achieved by polling the sensors or receiving events issued by the sensors themselves. An LIS does not require any knowledge of other LISs. To hide differences among the underlying locating systems, each LIS maps low-level positional information from each of the locating systems into information that is symbolic model of location. An LIS represents an entity's location in terms of the unique identifier of the sensor that detects the tag of the entity. We call each sensor's coverage a *cell*, as in the model of location studied in [8].

**Mechanism for Agent Discovery:** Each LIS discovers mobile agents bound to tags within its cells and 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 of the agent hosts in its current sub-network. It then waits for replies 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 the reply, 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; the 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 the following way. If the LIS has not received any replies 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 message with the identifier of the tag and the identifier of the cell to all agent hosts in its current sub-network.

**Navigation Service:** We will now explain how agents navigate to reach appropriate agent hosts. When an LIS detects the movement of a tag attached to a person or a 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) [20]. 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: the vendor and model class of the device (PC, PDA, phone, etc.), its screen size, the number of colors, CPU, memory, input devices, secondary storage, and presence/absence of speakers. 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 informs each agent about the profiles of agent hosts that are present in the cell and satisfies the requirements of the agent. The agents are then able to autonomously migrate to the appropriate hosts. The current implementation allows each agent to specify the preferable capabilities of agent hosts that it may visit as well as the minimal capabilities.

When there are multiple candidate destinations, each of the agents that is tied to a tag must select one destination on the basis of the profiles of the destinations. Also, when one or more cells geographically overlap, a tag may be in multiple cells at the same time; agents tied to that tag may then receive candidate destinations from multiple LISs. Our goal is to provide physical entities and places with computational functionality from locations 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 is not intended to move agents tied to the tag to hosts in different cells.

## 3.2 Agent Host

Each agent host has two forms of functionality: one for advertising its capabilities and the other for executing and migrating mobile agents. When a host receives a query with the identifier of a newly arriving tag from an LIS, it responds in one of the following three ways: (i) if the identifier in the message is equal 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.[1]

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 Java virtual machine, which hides differences between the platform architecture of 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 agents inside it and maintains the life-cycle state of each agent. When the life-cycle state of an agent changes, for example, 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 or release various resources, such as files, windows, and sockets, which it has previously captured. When a notification on the presence or absence of a tag is received from an LIS, the runtime system dispatches specific events to the agents that are tied to that tag and run inside it.

## 3.3 Mobile Agent Program

Each mobile agent is a collection of Java objects and is equipped with the identifier of the tag to which it is attached. It 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 another host, it can easily adjust its windows to the screen of the new host by using a compound document framework for the MobileSpaces system that was presented in our previous paper [15].

Next, we will explain the programming interface for our mobile agents. Every agent program must be an instance of a subclass of the abstract class `TaggedAgent` as follows:

[1] The current implementation assumes that LISs and agent hosts can be directly connected through a wireless LAN such as IEEE802.11b and thus does not support any multiple-hop query mechanisms, unlike mobile ad-hoc networking technology [12].

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

```
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) { ... }
  ....
}
```

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 `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 requirements that its destination hosts must satisfy by invoking the `setAgentProfile()` method, with the requirements specified as `apf`. The class provides a service method, `isConformableHost()`, which the agent uses to decide whether or not the capabilities of an agent host 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.

```
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. 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 the agent hosts that are present in the new cell. The `tagLeft` method is invoked after the tag is no longer in a cell.

### 3.4 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. The remainder of this section discusses some features of the current implementation.

**Locating Systems:** The current implementation of our framework supports two commercial locating systems: RF Code's Spider and Elpas's EIRIS. The former provides active RF-tags. Each tag has a unique identifier that periodically emits an RF-beacon that conveys an identifier (every second). The system allows us to explicitly control the omnidirectional range of each of the RF receivers to read tags within a range of 1 to 20 meters. The latter provides active infrared-tags, which periodically broadcast their identifiers through an infrared interface (every four seconds), like the Active Badge system [19]. Each infrared receiver has omnidirectional infrared coverage, which can be adjusted to cover distances within the range of 0.5 to 10 meters. Although there are many differences between the two locating systems, the framework minimizes the differences.

**Performance Evaluation:** Although the current implementation of the framework was not built for performance, we measured the cost of migration of an agent with a size of 3 Kbytes (zip-compressed) from a source host to the destination host recommended by the LIS. This experiment was performed with two LISs and two agent hosts, each of which was running on one of four computers (Pentium III-1GHz with Windows2000 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 an LIS had detected the presence of the tag of an agent was 380 msec; the cost of agent migration between two hosts over a TCP connection was 48 msec. The latency includes the cost of the following processes: UDP-multicasting of the identifier of the tags 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 of the agent; the migration of an agent from the source host to the destination host; unmarshaling of the agent; and security verification. We believe that this latency is acceptable for a location-aware system used in a room or building.

## 4 Initial Experience

To demonstrate the utility of the SpatialAgent framework, we developed several typical location-aware applications for mobile or ubiquitous computing settings.

### 4.1 Follow-Me Desktop Application

A simple application of the framework is a desktop *teleporting* system, like a *follow-me* application [3], within a richly equipped, networked environment such as a modern office. The system tracks the current location of a user and allows him or her to access his or her applications at the nearest computer as he or she moves around in the building. Unlike previous studies of such applications, our framework can migrate, not only the user interfaces of applications but also the applications themselves, to appropriate computers in the cell that contains the tag of the user. In our previous paper [15], we also developed a mobile window manager, which is a mobile agent that can carry its desktop applications as a whole to another computer and control the size, position, and overlap of the windows of the applications. Using the framework presented in this paper, the window manager and desktop applications can be automatically moved to and then executed at the computer that is in the current cell of the user and that has the resources required by the applications in the manner shown in Fig. 4.

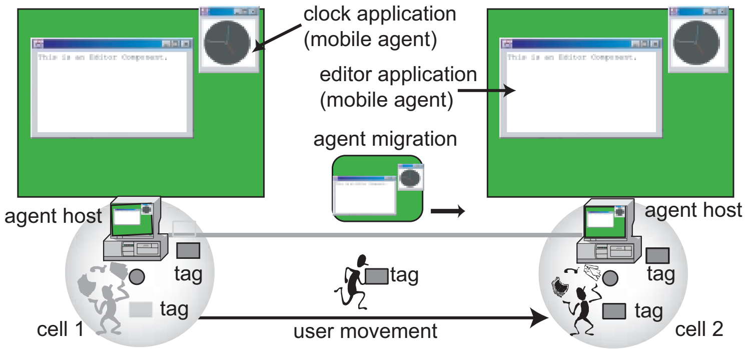

**Fig. 4.** Follow-Me Desktop Applications

## 4.2 User Navigation System

We also developed a user navigation system that assists visitors to a building. Several researchers have reported on other similar systems [2,4]. In this example, tags are distributed to several places within the building, such as its ceilings, floors, and walls; each visitor carries a wireless-LAN enabled tablet PC, which is equipped with a locating sensor to detect tags. It also includes an LIS and an agent host. The system initially deploys place-bound agents to invisible computers within the building. When a tagged position enters the cell of the moving sensor, 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. Next, it instructs the agent to migrate to its agent host and perform the agent's location-dependent services at the host. Fig. 5 shows a situation where a visitor with his/her tablet PC and sensor is roaming, first approaching place A and then place B. The system enables more than one agent tied to place A to move to the table PC. The agent returns to its home computer and other agent, which is tied to place B. It then moves to the tablet PC. Fig. 6 shows a place-bound agent displaying a map of its surrounding area on the screen of a tablet PC.

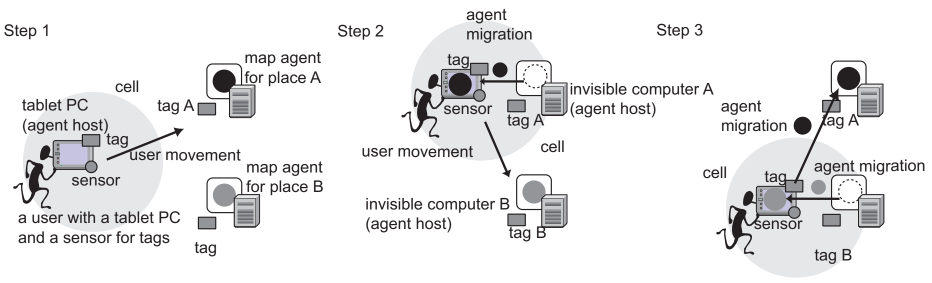

**Fig. 5.** The migration of an agent, which is attached to a place, to a visiting computer.

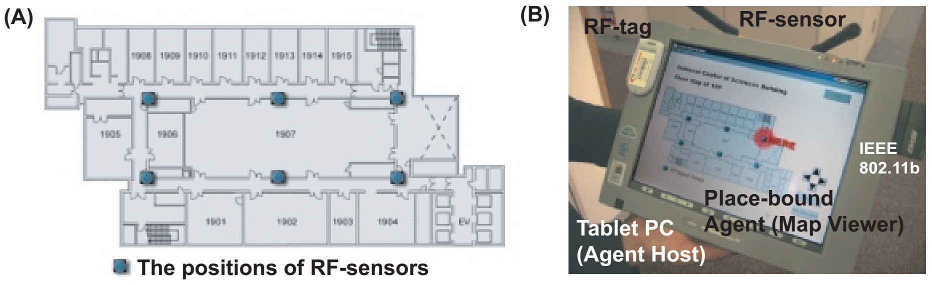

**Fig. 6.** (A) the positions of RF-tags on a floor (B) and a screen-shot of a map-viewer agent running on a table PC

## 4.3 Proactive Control of Home Appliances

We also used this framework to implement two prototype systems to control the lights in a room. Each light was equipped with a tag and was within the range covered by the sensor. In a previous project [11], we developed a generic server to control power outlets through a commercial protocol called the X10; in both the approaches we describe here the lights are controlled by switching their power sources on or off through the X10 protocol.

**User-aware Automatic Controller:** The first system provides proactive control of room lighting through a similar approach to that used by the EasyLiving project [1]. Our approach can autonomously turn the room lights on whenever a tagged user is sufficiently close to them. Suppose that each light is attached to a tag and is within the 3-meters coverage of the stationary sensor for the RF Code's Spider system. A tag attached to each of the lights is correlated with a mobile agent, which is a client of our X10-based server and is running on a stationary agent host in the room. When a tagged user approaches a light, an LIS in the room detects the presence of his/her tag in the cell that contains the light. Next, the LIS moves an agent that is bound to his/her tag to the agent host on which the light's agent is running. The user's agent then requests that the lights' agent to turn the light on through inter-agent communication.

**Location-aware Remote Controller:** The second system allows us to use a PDA as a remote controller for nearby lights. In this system, place-bound controller agents, which can communicate with X10-base servers to switch the lights on or off, are attached to the places that contain room lights. Each user has a tagged PDA, which supports an agent host with WindowsCE and a wireless LAN interface.[3] When a user with his/her 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 place, the agent automatically closes its user interface and returns to its home host.

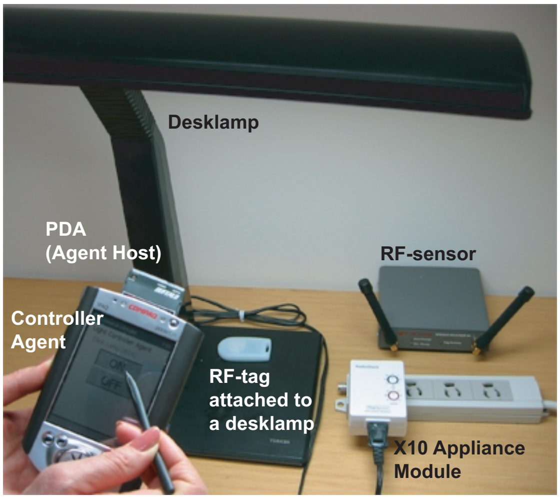

**Fig. 7.** Controlling a desk lamp from a PDA

## 5 Related Work

This section discusses several systems that have influenced various aspects of this framework, which seamlessly integrate two different approaches: ubiquitous and mobile computing.

First, we will compare some projects that support mobile users in ubiquitous computing environment with our framework. Research on smart spaces and intelligent environment is becoming increasingly popular at many universities and corporate research

[3] Since existing Java VMs for WindowsCE-based PDAs are lacking in terms of function and performance, the current implementation of this example uses a light-weight version of the MobileSpaces system.

facilities. Cambridge University/AT&T's Sentient Computing project [3] provides a platform for location-aware applications using infrared-based or ultrasonic-based locating systems in a building. Using the VNC system [13], 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, or it moves around. Although the platform 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. Since the applications must be executed in remote servers, the platform may have non-negligible interactive latency between the servers and the hosts that the user locally accesses. Recently, some Cambridge University researchers [9] have proposed a CORBA-based middleware, called LocARE, for the Sentient Computing project. The middleware can move CORBA objects to hosts according to the location of tagged objects. CORBA objects, however, are not always suitable for implementing user interface components. Microsoft's EasyLiving project [1] provides context-aware spaces, with a particular focus on the home and office. Computer-vision is used to track users within the spaces. The system can organize the dynamic aggregation of networked-enabled devices in a space and control devices according to the location of users.

There have also been several studies on enhancing context-awareness in mobile computing. HP's Cooltown [6] is an infrastructure for supporting context-aware services on portable computing devices. It is capable of automatically providing bridges between people, places, and things in the physical world and the web resources that are used to store information about them. The bridges that it forms allow users to access resources stored on the web via a browser, using standard HTTP communication. Although it would be an advantage in this system for users to be familiar with web browsers, all of the services available in the Cooltown system are constrained by the limitations of web browsers and HTTP. The NEXUS system [4], developed by Stuttgart University, offers a generic platform that supports location-aware applications for mobile users. Like the Cooltown system, users require PDA or tablet PC-like handheld devices, which are equipped with GPS-based positioning sensors and wireless communication. Applications that run on such devices, e.g., user-navigation, maintain a spatial model of the current vicinity of users and gather spatial data from remote servers. Unlike our approach, however, both of these approaches are not suited to supporting mobile users from stationary computers distributed in a smart environment.

Despite this, even though a number of mobile agent systems have been developed, few researchers have attempted to apply mobile agent technology to mobile and ubiquitous computing. Kangas [5] developed a location-aware augmented-reality system that enabled the migration of virtual objects to mobile computers, but only when the computer was located in particular spaces, like our framework. However, the system was not designed to move such virtual objects to ubiquitous computing devices. Hive [10] is a mobile agent-based middleware to control devices in ubiquitous computing environments, but it does not support any location-aware services.

## 6 Future Work

Since the framework presented in this paper is designed as a general-purpose framework, in future work we need to apply it to various applications as well as the three applications presented in this paper. Moreover, the MobileSpaces system, which is one basis of the framework, allows an application-specific service to be implemented as a collection of multiple agents rather than as a single agent. We are now developing a mechanism to divide an application-specific service into multiple mobile agents. For example, a mobile agent-based service may often require various I/O devices, such as a keyboard and speakers, but cannot find an agent host that has all of the devices. If there are two hosts, where one has a keyboard and another has speakers, the service should be able to be provided by the two in combination. The current mechanism to exchange information between LISs is not yet satisfactory. We therefore plan to develop a publish-subscribe system for the framework. We currently have an approach for building and managing configurable sensor networks [18]. Since it allows sensor nodes to be organized and configured according to the requirements of applications and changes in the physical world, it is useful in dynamically customizing our location information servers. We have also developed an approach to test context-aware applications on mobile computers [16]. We are interested in providing a methodology for testing applications based on the framework.

## 7 Conclusion

A novel framework for developing and managing location-aware applications in mobile and ubiquitous computing environments has been presented in this paper. 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. That is, it allows a mobile user to access its personalized services in a ubiquitous computing environment and provides location-dependent services to the user's portable computing device. The framework is decentralized. In addition, it is a generic platform independent of any higher-level applications and locating systems. We designed and implemented a prototype system for the framework and tested several practical applications.

## 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, September, 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, 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, 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 International Conference on Mobile Computing and Networking (MOBICOM'99), 249-255, 1999.
5. 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, 1999.
6. 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 Palo Alto, February, 2000.
7. B. D. Lange and M. Oshima, "Programming and Deploying Java Mobile Agents with Aglets", Addison-Wesley, 1998.
8. 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.
9. 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.
10. 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.
11. T. Nakajima, I. Satoh, and H. Aizu, "A Virtual Overlay Network for Integrating Home Appliances", Proceedings of International Symposium on Applications and the Internet (SAINT'2002), pp.246-253, IEEE Computer Society, January, 2002.
12. C. E. Perkins "Ad Hoc Networking", Addistion Wesley, 2001.
13. T. Richardson, Q, Stafford-Fraser, K. Wood, A. Hopper, "Virtual Network Computing", IEEE Internet Computing, Vol. 2, No. 1, 1998.
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, 2000.
16. I. Satoh, "Flying Emulator: Rapid Building and Testing of Networked Applications for Mobile Computers", in Proceedings of Conference on Mobile Agents (MA'2001), LNCS, Vol. 2240, pp.103-118, Springer, 2001.
17. M. Strasser and J. Baumann, and F. Holh, "Mole: A Java Based Mobile Agent System", Proceedings of 2nd ECOOP Workshop on Mobile Objects (eds. J. Baumann, C. Tschudin and J. Vitek), 1997.
18. T. Umezawa, I. Satoh, Y. Anzai, "A Mobile Agent-based Framework for Configurable Sensor Networks", to appear in International Workshop on Mobile Agents for Telecommunication Applications (MATA'2002), LNCS, Springer, October, 2002.
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, January, 1992.
20. World Wide Web Consortium (W3C), Composite Capability/Preference Profiles (CC/PP), http://www.w3.org/TR/NOTE-CCPP, 1999.