WIRELESS COMMUNICATIONS AND MOBILE COMPUTING
Wirel. Commun. Mob. Comput. 2003; **3**:411–423 (DOI: 10.1002/wcm.126)

SpatialAgents: integrating user mobility and program mobility in ubiquitous computing environments

Ichiro Satoh*,†

National Institute of Informatics
2-1-2 Hitotsubashi
Chiyoda-ku
Tokyo 101-8430
Japan

Summary

This paper presents a framework for the building of 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 navigational assistance system. Copyright © 2003 John Wiley & Sons, Ltd.

KEY WORDS

mobile agent
ubiquitous computing
location awareness
location sensor
user navigation
mobile computing
RFID tag
middleware

*Correspondence to: Ichiro Satoh, National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan.
†E-mail: ichiro@nii.ac.jp

1. Introduction

Ubiquitous computing and mobile computing are key areas in future computing. However, the two approaches have their own advantages and disadvantages. The concept of ubiquitous computing implies computation in elements that are contained in the environment rather than those 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 invading every aspect of life. This demonstrates the suitability of ubiquitous computing to provide environmental information and services. However, this approach is not suitable for providing multiplepurpose and personalized services, because the devices embedded in various items within the environment tend to have limited storage and processing capacity. They are thus incapable of internally maintaining a variety of software and profile databases on the 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 means that computing devices, for example, notebook-PCs, Personal Digital Assistants (PDAs), and wearable computers, are carried by the 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 their personalized forms, regardless of the locations of users. Each of these devices has been designed with the intent of staying with a particular user so that his/her profile can be maintained within it and can easily be evolved over time, without having to be transferred to 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 can be posed as polar opposites. We have attempted to mitigate the disadvantages of one 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 the 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 disadvantages in both approaches have been explored. AT&T's Sentient Computing [1], for example, proposed a so-called follow-me application to support the provision of personalized services in ubiquitous computing settings. HP's Cooltown [2] mobile computing devices such as PDAs and smart phones are attached to positioning sensors in order to give 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, which we developed using the framework (Section 4), and briefly review related work (Section 5). We briefly present some future issues (Section 6) and close with a summary (Section 7).

2. Approach

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

2.1. Locating Systems

Our goal is to offer a location-aware system in which spatial regions can be determined within a few square feet, which distinguishes one or more portions of a room or building. The framework itself is designed

to be independent of any particular infrastructure for location and is accompanied by more than one locating system. It determines the positions of objects by identifying the spatial regions that contain them. In general, such locating systems consist of radio frequency (RF) 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 entities that are automatically locatable.

The framework consists of two parts: (i) mobile agents and (ii) 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 provides a layer of indirection between the underlying location-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 themselves to.

2.2. Application-specific Services

Since each mobile agent is a programmable entity, the framework enables application-specific services, including user interface and application logic, to be implemented within mobile agents. 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 Remote Procedure Call (RPC)-based approaches, on which other existing approaches are often based, must have network latency between computers and remote servers. It can also directly access various equipment, which belong 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 the results of working, for example, 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 [1].
- Mobile and ubiquitous computers often have only limited resources, such as restricted levels of CPU power and memory. Mobile agents can help conserve these limited resources, since each agent only needs to be present at the computer while the computer needs the services provided by that agent.

The above advantages are unique in comparison with existing approaches, for example, Cambridge University's Sentient Computing system and HP's Cooltown systems. Moreover, this framework enables each mobile agent to be tied to a 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 must be equipped with its own tag and must advertise its profile information to LISs that detect the tag. The framework supports two types of linkage 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 or a nonelectronic 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 Figure 1 shows.
- The framework allows physical places to have their own agents that 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 to the visiting devices, where they provide the location-dependent services of the place as shown in Figure 2.

This framework permits a combination of both forms of linkage, while existing related approaches, such as Cambridge University'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

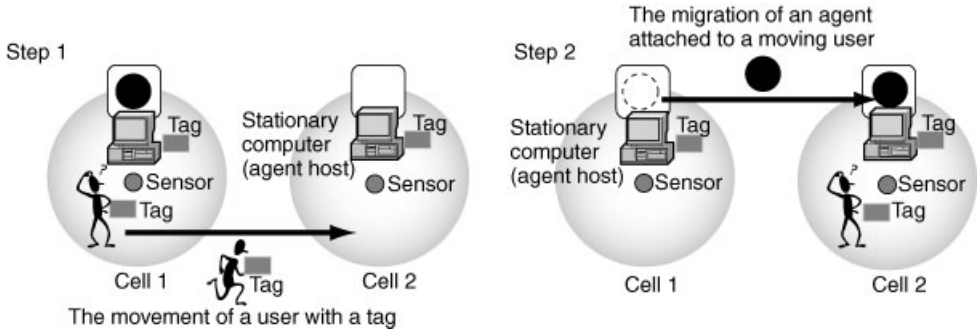

Fig. 1. Migration of an agent, which is attached to a moving entity, to a computer at the current location of the entity.

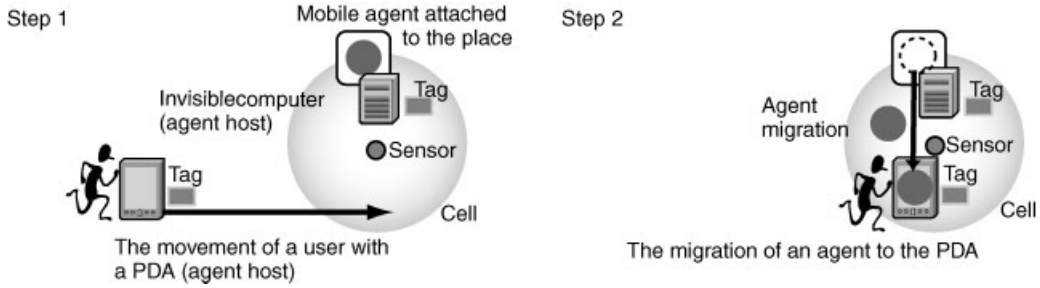

Fig. 2. Migration of an agent, which is attached to a particular place, to a computer visiting that place.

cover the spaces; instead, they can be placed near more than one agent host and the coverage areas of sensors can overlap.

2.4. Design Principles

In addition to accomplishing 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 from 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 its own configuration policy, whether or not it will migrate itself 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 individually connected to other 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 agents and agent hosts that are required; this mechanism was inspired by ad hoc mobile networking technology [3].

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 movements. Thus, the agents do not leak profile information on their users to other parties and they can interact with their mobile users in personalized form that has been adapted to respective individual users.

3. Design and Implementation

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

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 of its sensors. This is achieved by polling sensors or receiving the events issued by the sensors themselves. An LIS does not require any knowledge of other LISs. To conceal the differences among the underlying locating systems, each LIS maps low-level positional information from each of the locating systems into information in a 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 described by Leonhardt [4].

Mechanism for Agent Discovery: Each LIS discovers mobile agents that are bound to the 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 the tagged entity or place. When an LIS detects a new tag in a cell, it multicasts a query that contains the identity of the new tag and its own network address to all the agent hosts in its current subnetwork. It then waits for reply messages from the agent hosts. Here, there are two possible scenarios: the tag may be attached to an agent host or the tag may be attached to a person, place, or thing other than the 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, for example, its 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.

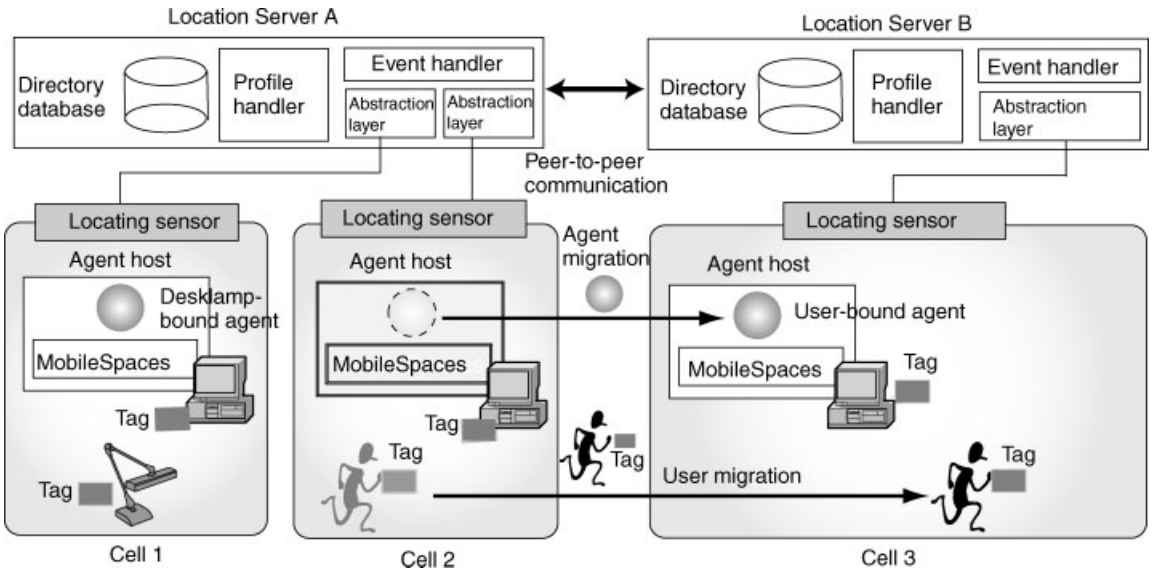

Fig. 3. Architecture of the SpatialAgent framework.

- 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 of each agent specify the capabilities of the agent hosts that the agent can visit and provide 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 manner discussed below. If the LIS does not have any reply messages from the agent hosts, it multicasts 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 subnetwork.

Navigation service: Next, we explain how agents navigate to reach appropriate 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) [5]. 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, presence/absence of loudspeakers, and so on. Each LIS is able to determine whether or not the device profile of each agent host satisfies the requirements of an agent by symbolical matching and quantitative comparison of properties. The LIS informs each agent about the profiles of agent hosts that are present in the cell and that satisfy 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 the 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 that are near them. Therefore, if there are no appropriate agent hosts in any of the cells in 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 another for executing and migrating mobile agents. 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* [6].[§] Each MobileSpaces runtime system is built on the Java virtual machine, which conceals 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 Transmission control protocol/Internet protocol (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, 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 various resources or release them, such as files, windows, or 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.

[‡] 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 [3].

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

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 other hosts, it can easily adjust its windows to the screen of the new host by using the compound document framework for the MobileSpaces system that was presented in our previous paper [7].

Next, we 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: class TaggedAgent extends Agent
      implements Serializable {
 2:   void go(URL url) throws
        NoSuchHostException { ... }
 3:   void duplicate() throws
        IllegalAccessException { ... }
 4:   void destroy() { ... }
 5:   void setTagIdentifier
        (TagIdentifier tid) { ... }
 6:   void setAgentProfile
        (AgentProfile apf) { ... }
 7:   URL getCurrentHost() { ... }
 8:   boolean isConformableHost
        (HostProfile hfs) { ... }
 9:   ....
10: }
```

We now 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 `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.

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.

```
 1: interface TaggedAgentListener
      extends AgentEventListener {
 2:   // invoked after creation at url
 3:   void agentCreated(URL url);
 4:   // invoked before termination
 5:   void agentDestroying();
 6:   // invoked before migrating to dst
 7:   void agentDispatching(URL dst);
 8:   // invoked after arrived at dst
 9:   void agentArrived(URL dst);
10:   // invoked after the tag
         arrived at another cell
11:   void tagArrived(HostProfile[]
         apfs, CellIdentifier cid);
12:   // invoked after the tag left
         rom the current cell
13:   void tagLeft(CellIdentifier cid);
14:   // invoked after an agent host
         arrived at the current cell
15:   void hostArrived(AgentProfile
         apfs, CellIdentifier cid);
16:   ....
17: }
```

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 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. Security and Privacy

Since agents carry the profile information about the entities, including people, and places that the agents are bound to, they are required to protect such information. Since the framework can be built on many Java-based mobile agent systems, it can directly use the security mechanism of the underlying mobile agent system and the Java virtual machine. Actually, the MobileSpaces system can encrypt agents that are to be encrypted before migrating them over a network and then decrypt them after they arrive at their destination. Moreover, since each mobile agent is just a programmable entity, it can explicitly encrypt its particular fields and migrate itself with

these fields and its own cryptographic procedure. The Java virtual machine can explicitly restrict agents to only access-specified resources to protect hosts from malicious agents. Although the current implementation cannot protect agents from malicious hosts,¶ the MobileSpaces system supports some authentication mechanisms for agent migration as mentioned in our previous paper [8] so that each agent host can only send agents to and only receive from the trusted hosts.

3.5. 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. 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 m. The other system provides active infrared tags, which periodically broadcast their identifiers through an infrared interface (every four seconds), like the Active Badge system [9]. Each infrared receiver has omnidirectional infrared coverage, which can be adjusted to cover distances in a range of 0.5 to 10 m. Although there are many differences between the two locating systems, the framework abstracts these differences away.

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 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 the LIS had detected the presence of the agent's tag was 380 ms and the cost of agent migration between two hosts over a TCP connection was 48 ms. The latency includes the costs 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 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 a building.

¶ This problem is beyond the scope of this paper.

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 Applications

A simple application of the framework is a desktop *teleporting* system, like a *follow-me* application [1], within a richly equipped, networked environment such as a modern office. The system tracks the current location of the user and allows him/her to access his/her applications at the nearest computer as he/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 user's tag. In our previous paper [7], we also discussed our development of a mobile window manager, which is a mobile agent and can carry its entire range of desktop applications to another computer and control the size, position, and overlap of the applications' windows. Using the framework presented in this paper, the window manager and desktop applications can automatically be moved to and then executed on the computer that is in the user's current cell, which has the resources required by the applications as shown in Figure 4.

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 [10,11]. In our system, tags are distributed to several places within a 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

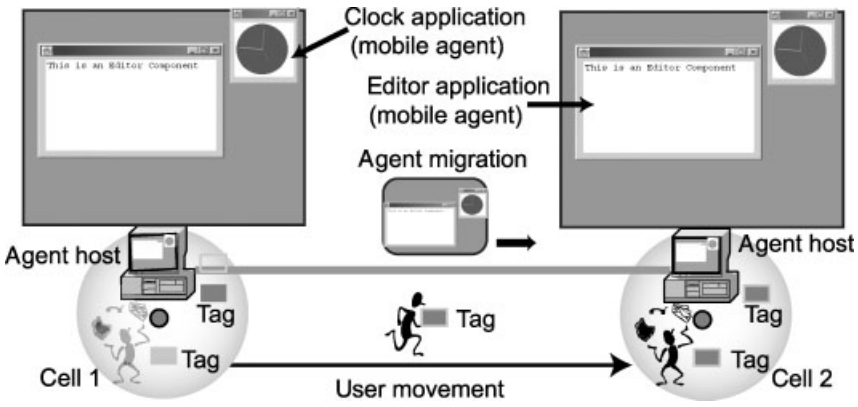

Fig. 4. Follow-me desktop applications.

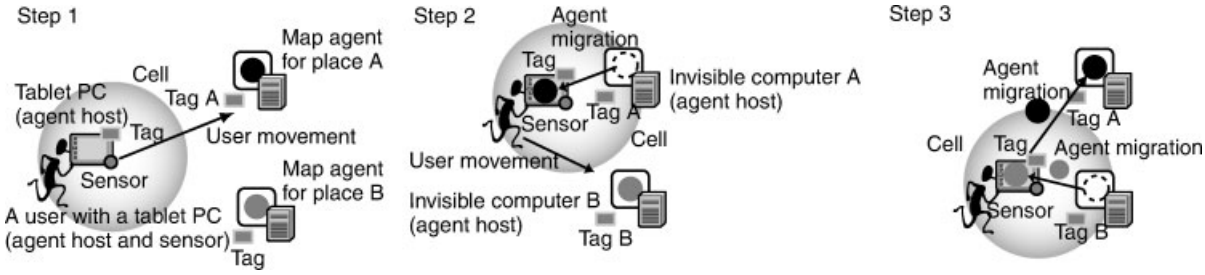

Fig. 5. The migration of an agent, which is attached to a place, to a visiting computer in the place.

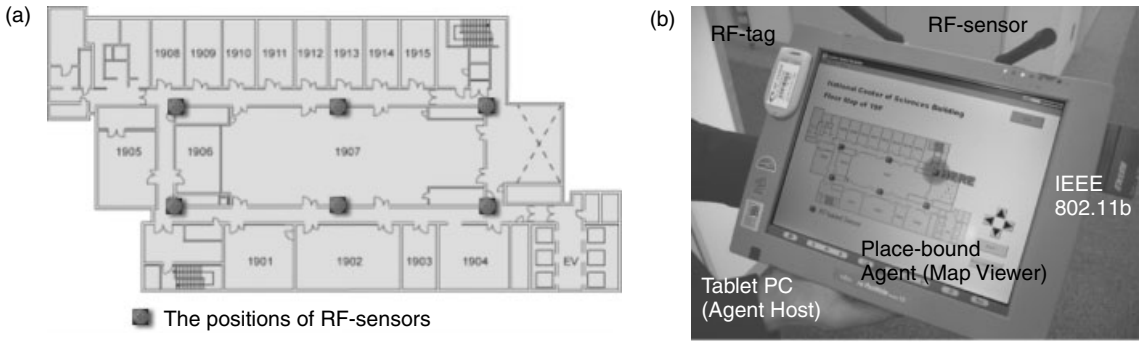

Fig. 6. (a) The positions of RF tags in a floor and (b) the screen shot of a map-viewer agent running on a tablet PC.

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 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. It then instructs the agent to migrate to its agent host and provide the agent's location-dependent services at the host. Figure 5 shows a situation in which a visitor with his/her tablet PC equipped with an RF-based 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 tablet PC; the agent then returns to its home computer and the other agents, which are tied to place B, move to the tablet PC. Figure 6 shows a place-bound agent displaying a map of its surrounding area on the screen of a tablet PC.

4.3. Proactive Control of Home Appliances

We also used this framework to implement two prototype systems to control electric lights in a room. Each

light is equipped with a tag and is within the coverage area of a sensor to detect tags. In a previous project [12], we developed a generic server to control power outlets through a commercial protocol called X10. In both the approaches that 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 used in the EasyLiving project [13]. Our approach can autonomously turn on room lights whenever a tagged user is sufficiently close to them. Suppose that each light is attached to a tag and is within 3 m of a stationary RF Code's Spider sensor. A tag attached to each of the lights is correlated with the mobile agent, which is our X10-based server's client and is running on the 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. The LIS then moves the 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 light's agent turns the light on through the interagent communication.

Location-aware remote controller: The second system allows us to use a PDA to remotely control nearby lights. In this system, place-bound controller agents, which can communicate with X10-based servers to switch lights on or off, are attached to places with room lights. As shown in Figure 7, each user has a tagged PDA, which supports the agent host with WindowsCE and a wireless LAN interface.[‖] When a user with a PDA visits the 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.

5. Related Work

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

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

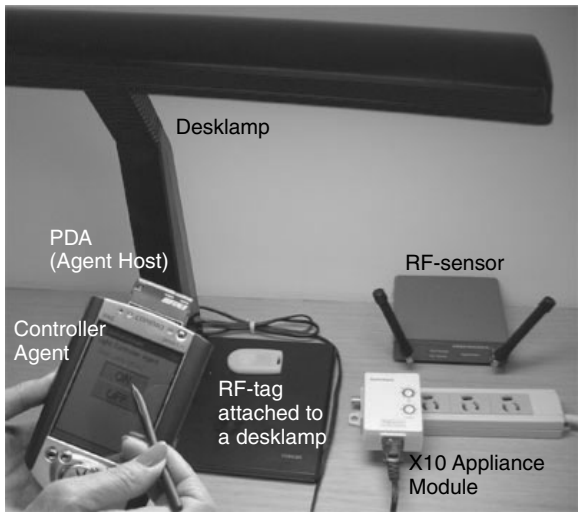

Fig. 7. Controlling a desk lamp from a PDA.

We compare several projects that support mobile users in a ubiquitous computing environment with our framework. Research on smart spaces and intelligent environments has become popular at many universities and corporate research facilities. Cambridge University Sentient Computing project [1] provides a platform for location-aware applications using infrared-based or ultrasonic-based locating systems in a building. Using the Virtual Network Comuting (VNC) system [14], 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 him/her while he/she moves around. Although the platform provides functionality similar 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 nonnegligible interactive latency between the servers and the hosts that the user accesses locally. Our framework, however, enables a user's applications, including user interfaces to be dynamically deployed and directly run on computers close to the user so that it can minimize temporal and spatial distances in interactions between the user and the applications. Recently, the project provided a CORBA-based middleware, called *LocARE* [15]. The middleware can move CORBA objects to hosts according to the location of tagged objects, but CORBA objects are not always suited to implementation on user interface components.

The EasyLiving project provides context-aware spaces, with a particular focus on home and office. It used mounted sensors such as stereo cameras on the room's walls and tracks the location and identity of people in the room. The system can dynamically aggregate network-enabled input/output devices, such as keyboards and mouses, even when they belong to different computers in the space. However, its management is centralized and it does not dynamically migrate software to other computers on the basis of the position of the users. 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, while our framework permits sensors to be mobile and scattered throughout in the space.

There have also been several studies on enhancing context awareness in mobile computing. HP's Cooltown [2] is an infrastructure that supports context-aware services on portable computing devices. It is capable of automatically providing bridges between people, places, and things in the physical world and 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 user familiarity with web browsers is an advantage in this system, all the services available in the Cooltown system are constrained by limitations of web browsers and HTTP. Our framework, however, is not limited in its web-based approach and can dynamically change mobile agent–based applications, including viewer programs for location-sensitive information based on the locations and requirements of users.

The NEXUS system [11], developed by Stuttgart University, offers a generic platform that supports location-aware applications for mobile users. Like the Cooltown system, users require a PDA or tablet PC, which is equipped with Global Positioning System (GPS)-based positioning sensors and wireless communication. Applications that run on such devices, for example user navigation, maintain a spatial model of the current vicinity of users and gather spatial data from remote servers. Unlike our approach, however, neither Cooltown nor NEXUS is suitable to support mobile users through stationary computers distributed in a smart environment.

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 [16] developed a location-aware augmented-reality system that enables the migration of virtual objects to mobile computers, but only when the computer is in a particular space, like our framework. However, the system is not designed to move such virtual objects to ubiquitous computing devices. Hive [17] is a mobile agent–based middleware to control devices in ubiquitous computing environments, but it does not support location-aware services.

Several researchers have explored location-sensitive servers like our LIS. Their location models can be classified into two types: spatial models based on the concrete geographical coordinates of objects and spatial models based on the geographical containment between objects. For example, the EasyLiving project provides a geometric model based on the former approach, so it accurately represents the physical relationships between entities in the world. Leonhardt [4] developed a location-tree model based on the latter approach and used location-aware directory servers. Our framework is based on a symbolic location model like the geographical containment model. However, it is unique to existing work in having the ability to dynamically manage spatial models. That is, it provides a demand-driven mechanism that discovers the locations of agent hosts and agents, because it permits all its elements, such as hosts and sensors, to be mobile in and dynamically added to or removed from a space.

We described an approach for developing location-aware mobile agents in a previous paper [18]. The approach allows mobile agents to follow their users as they move, like the framework presented in this paper. However, this approach allows the positions of the users to be detected through a computer vision technique and it maintains a geographical model of the environment, including the positions of the users. On the other hand, our present framework uses RF or IR sensors to detect their presence and it maintains a symbolic location model in the sense that it can only detect the presence of tagged entities that are within the coverage of the sensors. Also, since the target of the previous approach was to support mobile users from stationary computers in a ubiquitous computing environment, it could not support mobile users from portable computing devices, whereas our framework can support both ubiquitous and mobile computing environments. Another paper [19] presented an early prototype of the present framework, but did not provide four linkages between physical and virtual worlds as described in the second section of this paper.

6. Future Work

Since the framework presented in this paper is general-purpose, in future work we need to apply it to specific applications as well as to the three applications presented in this paper. Moreover, the MobileSpaces system, which is the basis of this framework, allows application-specific services to be implemented as a collection of multiple agents rather than a single agent. We are now developing a mechanism that enables an application-specific service to be divided into multiple mobile agents. For example, a mobile agent–based service may often require various I/O devices, such as keyboards and speakers, but cannot locate an agent host that has all of these. If there are two hosts, where one has a keyboard and the other has speakers, the service can be provided by the two hosts in combination. The current mechanism for the exchange of information between LISs is not satisfactory. We therefore plan to develop a publish–subscribe system for the framework. We have an approach in mind that would enable the construction and management of configurable sensor networks [20]. Since this approach allows sensor nodes to be organized and configured according to application requirements and changes in the physical world, it would be useful in the dynamic customization of our location information servers. We have also developed an approach to the test of context-aware applications on mobile computers [21]. We are interested in developing a methodology that would test applications that were based on the framework.

7. Conclusion

A novel framework for the development and management of 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 a 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 have designed and implemented a prototype system of the framework and demonstrated its effectiveness in several practical applications.

References

1. Harter A, Hopper A, Steggeles P, Ward A, Webster P. The anatomy of a context-aware application. In *Proceedings of Conference on Mobile Computing and Networking (MOBICOM '99)*, ACM Press, 1999, pp. 59–68.
2. Kindberg T, Barton J, Morgan J, Becker G, Caswell D, Debaty P, Gopal G, Frid M, Krishnan V, Morris H, Schettino J, Serra B. People, Places, Things: Web Presence for the Real World. Technical Report HPL-2000-16, Internet and Mobile Systems Laboratory: HP Laboratories, Palo Alto, CA, 2000.
3. Perkins CE. *Ad Hoc Networking*. Addison-Wesley: Reading, MA, 2001.
4. Leonhardt U, Magee J. Towards a general location service for mobile environments. In *Proceedings of IEEE Workshop on Services in Distributed and Networked Environments*, IEEE Computer Society, 1999, pp. 43–50.
5. World Wide Web Consortium (W3C). *Composite Capability/Preference Profiles (CC/PP)*. http://www.w3.org/TR/NOTE-CCPP, 1999.
6. Satoh I. MobileSpaces: a framework for building adaptive distributed applications using a hierarchical mobile agent system. In *Proceedings of International Conference on Distributed Computing Systems (ICDCS 2000)*, IEEE Computer Society, 2000, pp. 161–168.
7. Satoh I. MobiDoc: a framework for building mobile compound documents from hierarchical mobile agents. In *Proceedings of Symposium on Agent Systems and Applications/Symposium on Mobile Agents (ASA/MA 2000)*, Lecture Notes in Computer Science, Springer, Vol. 1882, 2000, pp. 113–125.
8. Satoh I. Network processing of mobile agents, by mobile agents, for mobile agents. In *Proceedings of Workshop on Mobile Agents for Telecommunication Applications (MATA 2001)*, Lecture Notes in Computer Science, Springer, Vol. 2164, 2001, pp. 81–92.
9. Want R, Hopper A, Falcao A, Gibbons J. The active badge location system. *ACM Transactions on Information Systems* 1992; **10**(1): 91–102.
10. Cheverst K, Davis N, Mitchell K, Friday A. Experiences of developing and deploying a context-aware tourist guide: the GUIDE project. In *Proceedings of Conference on Mobile Computing and Networking (MOBICOM 2000)*, ACM Press, 2000, pp. 20–31.
11. Hohl F, Kubach U, Leonhardi A, Rothermel K, Schwehm M. Next century challenges: nexus—an open global infrastructure for spatial-aware applications. In *Proceedings of International Conference on Mobile Computing and Networking (MOBICOM '99)*, ACM Press, 1999, pp. 249–255.
12. Nakajima T, Satoh I, Aizu H. A virtual overlay network for integrating home appliances. In *Proceedings of International Symposium on Applications and the Internet (SAINT 2002)*, IEEE Computer Society, 2002, pp. 246–253.
13. Brumitt BL, Meyers B, Krumm J, Kern A, Shafer S. EasyLiving: technologies for intelligent environments. In *Proceedings of International Symposium on Handheld and Ubiquitous Computing*, 2000, 12–27.
14. Richardson T, Stafford-Fraser Q, Wood K, Hopper A. Virtual network computing. *IEEE Internet Computing* 1999; **2**(1): 33–38.
15. Lopez de Ipina D, Lo S. LocALE: a location-aware lifecycle environment for ubiquitous computing. In *Proceedings of Conference on Information Networking (ICOIN-15)*, IEEE Computer Society, 2001.

16. Kangas K, Roning J. Using code mobility to create ubiquitous and active augmented reality in mobile computing. In *Proceedings of Conference on Mobile Computing and Networking (MOBICOM '99)*, ACM Press, 1999, pp. 48–58.
17. Minar N, Gray M, Roup O, Krikorian R, Maes P. Hive: distributed agents for networking things. In *Proceedings of Symposium on Agent Systems and Applications/Symposium on Mobile Agents (ASA/MA '99)*, IEEE Computer Society, 1999.
18. Tanizawa Y, Satoh I, Anzai Y. A mobile agent framework for ubiquitous computing environments. *Information Processing Society Journal* 2002; **43**(12): 3774–3784 (in Japanese).
19. Satoh I. Physical mobility and logical mobility in ubiquitous computing environments. In *Proceedings of Conference on Mobile Agents (MA '02)*, Lecture Notes in Computer Science, Springer, Vol. 2535, 2002, pp. 186–202.
20. Umezawa T, Satoh I, Anzai Y. A mobile agent-based framework for configurable sensor networks. In *Proceedings of International Workshop on Mobile Agents for Telecommunication Applications (MATA 2002)*, Lecture Notes in Computer Science, Springer, Vol. 2521, 2002, pp. 128–140.
21. Satoh I. Flying emulator: rapid building and testing of networked applications for mobile computers. In *Proceedings of Conference on Mobile Agents (MA 2001)*, Lecture Notes in Computer Science, Springer; Vol. 2240, 2001, pp. 103–118.

Author's Biography

Ichiro Satoh received his B.E., M.E., and Ph.D. degrees in computer science from Keio University, Japan, in 1996. From 1996 to 1997, he was a research associate in the Department of Information Sciences, Ochanomizu University, Japan, and from 1998 to 2000 he was an associate professor in the same department. Since 2001, he has been an associate professor in National Institute of Informatics, Japan. He was also a researcher of Japan Science and Technology Corporation from 1999 to 2001. His current research interests include distributed and mobile computing. He received IPSJ paper award, IPSJ Yamashita SIG research award, and JSSST Takahashi research award. He is a member of six learned societies, including ACM and IEEE.