International Journal on Digital Libraries (2006)
DOI 10.1007/s00799-006-0006-1

REGULAR PAPER

**Ichiro Satoh**

# Location-based services in ubiquitous computing environments

**Abstract** This paper presents a framework for providing dynamically deployable services in ubiquitous computing settings. The goal of the framework is to provide people, places, and objects with computational functionalities to support and annotate them. Using RFID-based tracking systems, the framework detects the locations of physical entities, such as people or things, and deploys services bound to the entities at proper computing devices near where they are located. It enables location-based and personalized information services to be implemented as mobile agents and operated at stationary or mobile computing devices, which are at appropriate locations, even if the services do not have any location-information. This paper presents the rationale, design, implementation, and applications of our prototype infrastructure.

**Keywords** Ubiquitous computing · Mobile agent · Location-based services · Location-sensing system · Middleware

## 1 Introduction

As Mark Weiser envisioned [21], a goal of ubiquitous computing is to provide various services by making multiple computers available throughout the physical environment, but, in effect, making them invisible to the user. Another goal of ubiquitous computing is for it to integrate the physical world with cyberspace. Actually, perceptual technologies have made it possible to detect the presence or positions of people and any other object we care to think about. Context-awareness, in particular user-awareness and location-awareness, is becoming an essential feature of services that assist our everyday lives in ubiquitous and mobile computing environments.

I. Satoh (✉)
National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan
E-mail: ichiro@nii.ac.jp

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

This paper presents a framework for deploying and operating location-based or personalized information services to solve these problems and this is based on two key ideas. The first is to introduce mobile agent technology [7] as a mechanism for the deployment of services. Since many computing devices in ubiquitous and mobile computing environments only have limited resources, they cannot provide all services required due to limited computational resources, even if they are at suitable locations. Therefore, our framework provides an infrastructure for dynamically deploying service-provider agents to support services at computers that need the services. The second idea is to separate application-specific services from the infrastructure. Since each mobile agent is a programmable entity, the framework enables application-specific services, including user interfaces and application logic, to be implemented within mobile agents. Using mobile agents makes the framework independent of applications, because application-specific services are implemented within mobile agents instead of the infrastructure. Since the framework is responsible for automatically deploying mobile agents at appropriate computers, they can provide their services without any location-information. It delegates agent migration to its underlying mobile agent platform to execute and migrate mobile agents over a network. Nevertheless, it is available with various existing mobile agent platforms because it has been designed to be independent particular platforms.

In the remainder of this paper, we describe our basic ideas (Sect. 2) and the design of our framework (Sect. 4). We explain how information services can be bound to physical entities and places (Sect. 5) and describe the current implementation of the framework (Sect. 6). We also discuss our experience with several applications, which we developed with the infrastructure (Sect. 6.2). We briefly review related work (Sect. 7). We also provide a summary and discuss some future issues (Sect. 8). We describe programming models (Appendix).

## 2 Approach

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

### 2.1 Example scenarios

To outline the goals of this framework, we present two typical scenarios for it. The first provides personalized services to users without portable computers. When a user enters an unfamiliar building, he/she may lose his/her way or may not able to find any cafes there. The framework provides his/her personal agent that can assist him/her in his/her personal form any location. This is because these agents can migrate between computers and be executed on public terminals on streets or electric displays in front of cafes or restaurants. As the user moves, they follow the user's movements. When the user stops in front of the electric display of a cafe, the agent migrates to a computer in front of the cafe and displays the list of his/her favorite coffees on the screen of the computer. The second assumes that the user is carrying a PDA. Suppose that a room has have many electric lights. When he/she only wants to turn the electric lights near him/her on, he/she may occasionally know which switches on the wall-mounted central control panel controls the lights. This framework enables his/her PDA to be used as a universal remote controller. When a user goes near the lights, the framework displays a graphical user interface for the lights on the screen of the PDA for him/her to control them. The interface can turn the lights on or off through a stationary agent running on the control panel. When he/she leaves from the vicinity of the lights, the framework automatically closes the interface from the PDA, because of the device's memory small.

The first scenario is where its services are contained in the environment rather than carried on the person. The second is where services are carried by users rather than contained within the environment. Existing approaches aim at supporting one of either of these scenarios, whereas our framework can support the both scenarios with a unified approach.

### 2.2 Location sensing systems

The framework offers a location-aware system where spatial regions can be determined within a few square feet, which distinguishes between one or more portions of a room or building. Existing location-based services are typically tailored to a particular type of tracking or positioning technology, such as GPS. The current implementation of the framework uses RFID (radio frequency identification) technology as an alternate approach to locate objects. An RFID system consists of RFID readers (so-called sensors or receivers), which detect the presence of small RF transmitters, often called *tags*. Advances in wireless technology enable passive RFID tags to be scanned over a few meters. For example, the Auto-ID center (currently called EPCGlobal) [1] and its sponsors are working to develop flexible tags and readers operating at ultra-high frequency (915 MHz in the US, 868 MHz in the EU, and 950 MHz in Japan). It expects that RFID tags will cost around 5 cents when produced in bulk and RFID readers will cost around a 100 dollars in volume. The framework assumes that physical entities, including people, computing devices, and places will be equipped with RFID tags so that they will be automatically locatable.

The framework reads to provide a unified model for spatial information to hide the differences between the underlying location-sensing systems from applications as much as possible. Spatial information should also be bound within the requirements of an application that uses it to avoid unnecessary exposure of details on underlying tracking and positioning systems. Therefore, the model is based on symbolic location. This is because the framework aims at building location-aware applications for annotating and supporting people, objects, and places and such applications are often associated with semantic and structural spaces, such as buildings, rooms, and portions of a room or building, rather than geometric locations.

### 2.3 Dynamically deployable services

Suitable services should be operated on suitable computing devices in the sense that the services are both required according to the location of users and their associated contexts and the locations and capabilities of the devices can satisfy the requirements of the services. However, most ubiquitous and mobile computing devices often have only limited resources, such as restricted levels of CPU power and amounts of memory. As a result, even if a device is at a suitable location for the required service to be provided, the device may not be available because of a lack of software or capabilities, such as input or output facilities, to execute the software. Various kinds of infrastructure have been used to construct and manage location-aware services. However, such infrastructures have mostly focused on a particular application, such as user navigation. To solve these limitations, our framework uses mobile agent technology because the technology has the following advantages for ubiquitous and mobile computing settings:[1]

- Each mobile agent can travel from computer to computer under its own control. When a mobile agent moves to another computer, both the code and the state of the agent is transferred to the destination. Each agent only needs to be present on the device at the time when the device is required to offer the services provided by that agent. Therefore, mobile agents can help to conserve the limited resources of computing devices. After arriving at its destination, a mobile agent can continue working without losing the results, e.g., the content of instance variables in the agent's program, at the source computers.
- Since each mobile agent is a programmable entity, the framework enables application-specific services, including the user interface and application logic, to be implemented within mobile agents. It then separates application-specific services from itself. Therefore, it can be a general infrastructure for a variety of location-aware services. It can also directly access various equipment belonging to that device as long as the security mechanisms of the device permit this.

The infrastructure presented in this paper enables a physical entity and place to spatially bind with one or more mobile agent-based services. These services annotate and support the entities or places in the sense that they can be dynamically deployed at stationary and mobile computing devices that are near or within the locations of the entities and places. Therefore, the services can easily be customized to be person- and location-dependent. They can directly interact with their users, whereas other existing approaches, e.g., remote procedure calls and web-based interaction can be seriously affected by network latency between client-side and service-side computers.

### 2.4 Architecture

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

When an LIS detects a moving tag, it notifies mobile agents attached to it about the network addresses and capabilities of the candidate hosts that are near its location. Each of these agents selects one host from the candidate agent hosts recommended by the LIS and migrate to the selected host. The capabilities of a candidate host do not always satisfy all the requirements of an agent. No agent has to have any information about the network addresses and locations of devices, which it may migrate to. This framework assumes that each agent itself should decide, on the basis of to its own configuration policy, whether or not it will migrate itself to the destination and adapt itself to the destination's capabilities.

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

## 3 Design

This section presents the design of this framework and describes a prototype implementation of it. Figure 1 outlines the basic structure of the framework.

### 3.1 Location information server

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

#### *3.1.1 RFID-based location model*

This framework represents the locations of objects with symbolic names to specify the sensing ranges of RFID readers, instead of geographical models. Each LIS manages more than one RFID reader that detect the presence of tags and maintain up-to-date information on the identities of those that are within the zone of coverage. This is achieved by polling the readers or receiving events issued by them. An LIS does not require any knowledge on other LISs, but it needs to be able to exchange its information with

[1] Some applications can be constructed by means of a mobile code approach rather than mobile agent approach. However, since the former can be treated as a subset of the latter, our framework should support a more general approach, i.e., using mobile agents.

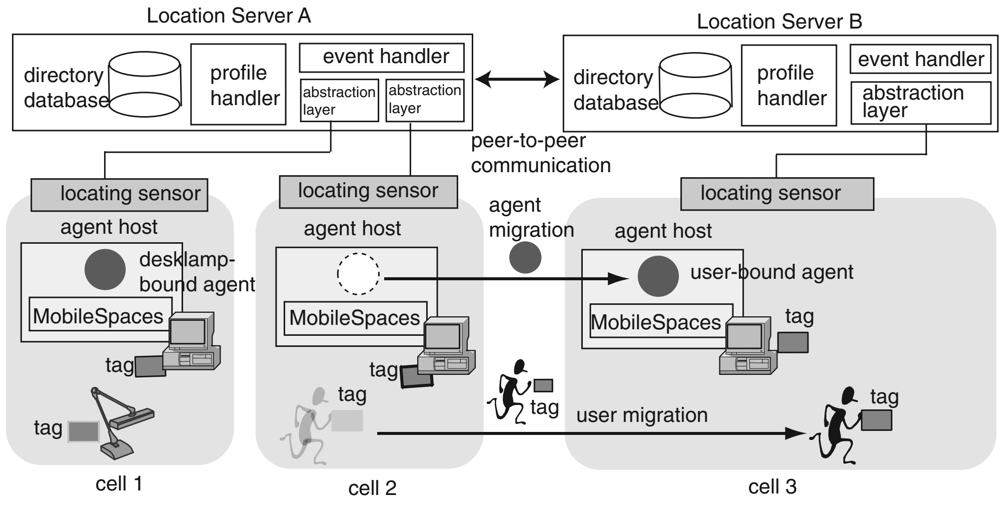

**Fig. 1** Architecture

others through multicast communication. To hide the differences between underlying locating systems, each LIS maps low-level positional information from the other LISs into information in a symbolic model of location. An LIS represents an entity's location in symbolic terms of the RFID reader's unique identifier that detects the entity's tag. Each RFID reader's coverage is called a *cell*, as in the models of location reported by several other researchers [8]. Multiple RFID readers in the framework do not have to be neatly distributed in spaces such as rooms or buildings to completely cover the spaces; instead, they can be placed near more than one agent host and reader coverage can overlap.

### *3.1.2 Location management*

Each LIS is responsible for discovering mobile agents bound to tags within its cells. Each maintains a database in which it stores information about each of the agent hosts and each of the mobile agents attached to a tagged entity or place. When an LIS detects a new tag in a cell, the LIS multicasts a query that contains the identity of the new tag and its own network address to all the agent hosts in its current subnetwork. It then waits for reply messages from the agent hosts. Here, there are two possible situations: 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, the newly arriving agent host will send its network address and device profile to the LIS; the profile describes the capabilities of the agent host, e.g., input devices and screen size. After receiving a reply message, the LIS stores the profile in its database and forwards the profile to all agent hosts within the cell.
- In the second, 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 a manner that will be discussed later. If the LIS has no reply messages from the agent hosts, it can multicast a query message to other LISs. When the absence of a tag is detected in a cell, each LIS multicasts a message with the identifier of the tag and the identifier of the cell to all agent hosts in its current subnetwork. Figure 2 shows the sequence for migrating an agent to a suitable host when an LIS detects the presence of a new tag.

### *3.1.3 Location-dependent deployment of agents*

We will now explain how the framework deploys agents at suitable agent hosts. When an LIS detects the movement of a tag attached to a person or thing to a cell, it searches its database for agent hosts that are present in the current cell of the tag. It also selects candidate destinations from the set of agent hosts within the cell, according to their respective capabilities. The framework offers a language based on CC/PP (composite capability/preference profiles) [22]. The language is used to describe the capabilities of agent hosts and the requirements of mobile agents in an XML notation. For example, a description contains information on the following properties of a computing device: vendor and model class of the device (i.e, PC, PDA, or phone), its screen size, the number of colors, CPU, memory, input devices, secondary storage, and the presence/absence of loudspeakers. The framework also allows each agent to specify the preferable capabilities of agent hosts that it may visit as well as the minimal capabilities in a CC/PP-based notation. Each LIS is able to determine whether or not the device profile of each agent host satisfies the requirements of an agent by symbolically matching and quantitatively comparing properties.

The LIS then unicasts a navigation message to each of the agents that are bound to the tagged entities or places, where the message specifies the profiles of those agent hosts that are present in the cell and satisfy the requirements of

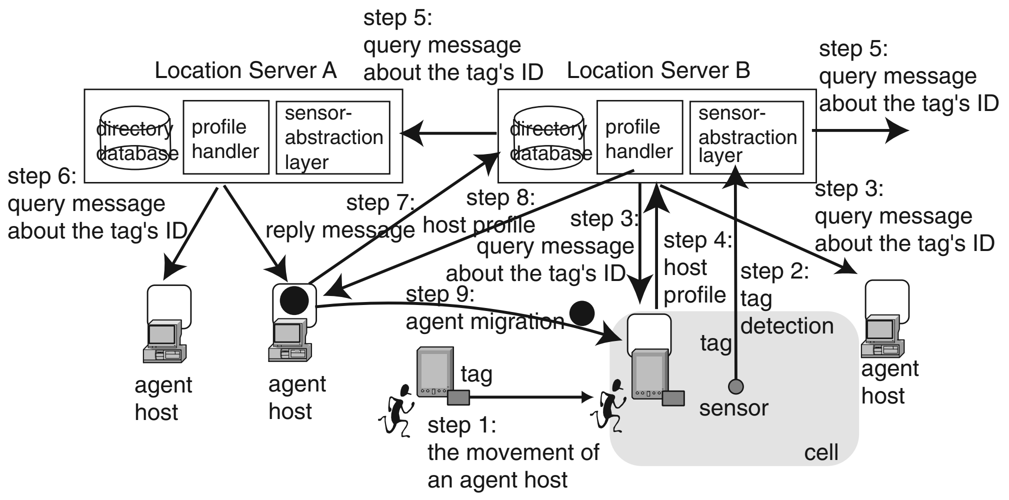

**Fig. 2** Agent discovery and deployment

the agent. The agents are then able to autonomously migrate to the appropriate hosts. When there are multiple candidate destinations, each of the agents that is tied to a tag must select one destination based on their profiles. When one or more cells geographically overlap, a tag may be in multiple cells at the same time and agents tied to that tag may then receive candidate destinations from multiple LISs. However, since the message includes the network address of the LIS, the agents can explicitly ask it about the cell ranges. For example, in the first scenario in Sect. 2 the assistant agent is bound to the RF-tag that the user has. When an LIS detects the tag is moving, it instructs the agent to migrate to a computer near the current position of the user. In the second scenario, an LIS detects the presence of the tag bound to a visiting PDA and the tags bound to the lights and then instruct agents controlling the lights to migrate to the PDA.

Our goal is to provide physical entities and places with computational functionality from locations that are near them. Therefore, if there are no appropriate agent hosts in any of the cells at which a tag is present but there are some agent hosts in other cells, the current implementation of our framework forces agents tied to the tag to move to hosts in different cells.

## 3.2 Mobile agent-based service-provider

The framework encapsulates application-specific services into mobile agents so that it is independent of all applications and can support multiple services. In the appendix to this paper, each mobile agent is constructed as a collection of Java objects and is equipped with the identifier of the tag to which it is attached.[2] Each is a self-contained program and is able to communicate with other agents and external systems. 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. For example, in the first scenario in Sect. 2, the assistant agent stores its user profile in its inner database, so that it can encapsulate privacy information inside it. A mobile agent may also have one or more graphical user interfaces for interaction with its users. When such an agent moves to other hosts, it can easily adjust its windows to the new host's screen by using the compound document framework for the MobileSpaces system that was presented in our previous paper [14].

## 3.3 Agent host

Each agent host must be equipped with a tag. It has two forms of functionality: the first for advertising its capabilities and the second for executing and migrating mobile agents. The current implementation assumes that LISs and agent hosts can be directly connected through a wired LAN e.g., Ethernet or a wireless LAN e.g., IEEE802.11a/b/g. 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 [13].[3] Each MobileSpaces runtime system is built on the Java virtual machine, which conceals differences between the platform architecture of the source and destination hosts, such as the operating system and hardware. Each of the runtime systems moves agents to other agent hosts over a TCP/IP connection. The runtime system governs all the

[2] Appendix describes programming interfaces of agents.

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

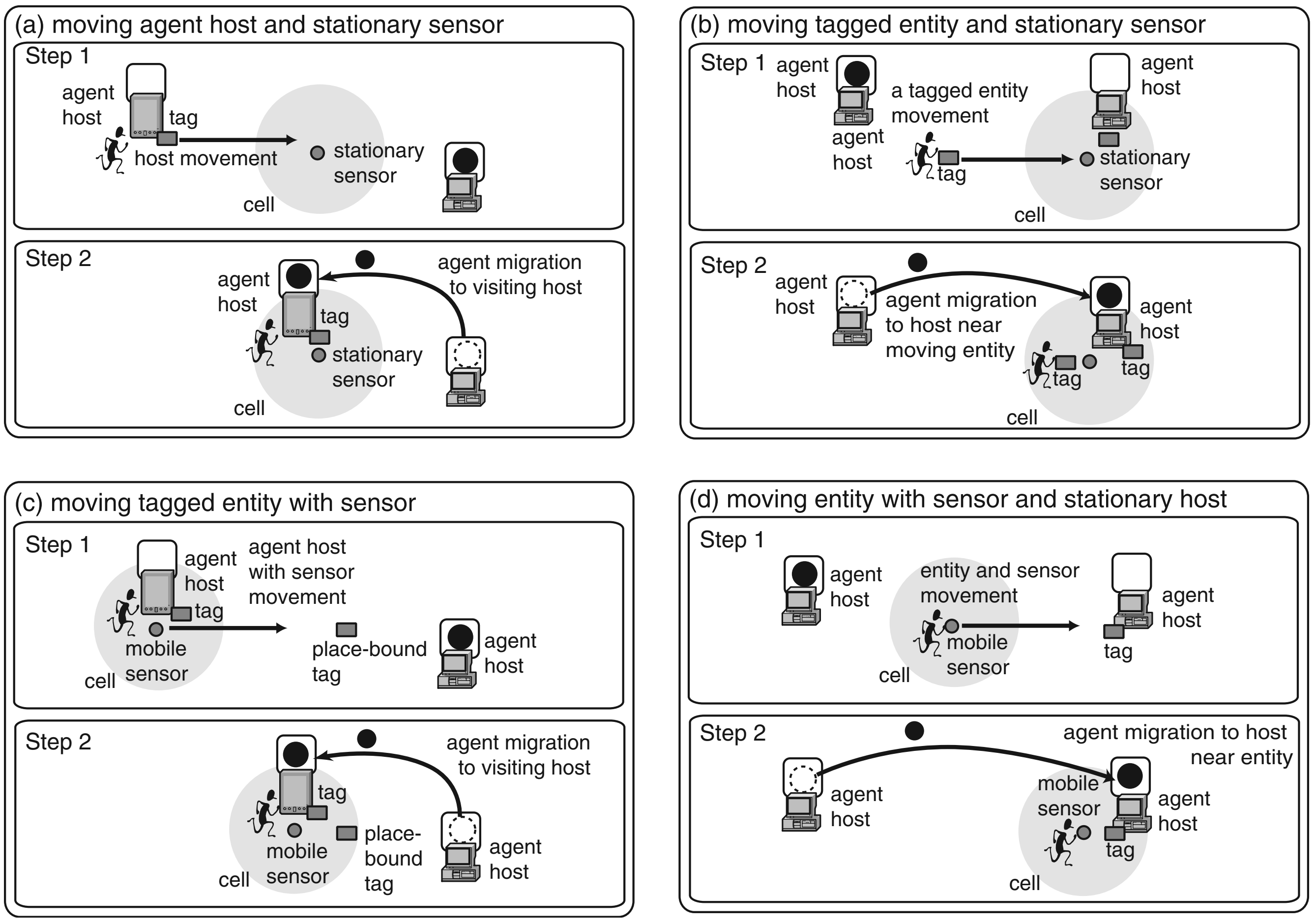

**Fig. 3** Linkages between physical and logical entities

agents inside it and maintains the life-cycle state of each agent. When the life-cycle state of an agent changes, e.g., when it is created, terminates, or migrates to another host, the runtime system issues specific events to the agent. This is because the agent may have to acquire various resources or release them, such as files, windows, or sockets, which it had previously captured. When a notification on the presence or absence of a tag is received from a LIS, the runtime system dispatches specific events to the agents that are tied to that tag and these run inside it.

## 4 Linkages physical worlds to logical worlds

The framework does not have to distinguish between mobile and stationary computing devices or between mobile and stationary location sensing systems. Therefore, it can support the following four types of linkages between a physical entity, such as a person or object, or place, and more than one mobile agent. Figure 3 illustrates the four linkages when entities and agent hosts are attached to RFID tags.

- Figure 3a shows that a moving entity carries an RFID-tagged agent host and a space contains a place-bound RFID tag and an RFID reader. When the reader detects the presence of the RFID tag that is bound to the agent host, the LIS instructs the agents attached to the tagged place to migrate to the visiting agent host to offer the location-dependent services of the place.
- Figure 3b shows that an RFID-tagged agent host and an RFID reader have been allocated. When an RFID-tagged moving entity enters the coverage area of the reader, the LIS instructs the agents attached to the entity to migrate to the agent host within the same coverage area to offer the entity-dependent services for the entity.
- Figure 3c shows that a moving entity carries a reader and agent host and a space contains a place-bound RFID tag. When the entity moves near the tag and the reader detects the presence of the tag within its coverage area, the LIS instructs the agents attached to the tagged place to migrate to the visiting agent host to offer the location-dependent services of the place.
- Figure 3d shows that an entity carries an RFID reader and a space contains a place-bound RFID tag and an RFID-tagged agent host. When the entity moves and the reader detects the presence of an agent host's tag within its coverage area, the LIS instructs the agents attached to the moving entity to migrate to the agent host within the

same coverage area to offer services dependent on the entity.

Note that the above linkages are independent of the underlying locating systems. Therefore, they are available in various sources of location information, e.g., GPS, local wireless networks, and cellular networks. Existing location-aware systems can only support one of the above linkages, whereas our infrastructure does not have to distinguish between them and can synthesize them seamlessly. For example, the linkage in Fig. 3a corresponds to the *person tracking display* approach in the EasyLiving project [2], the linkage shown in Fig. 3b corresponds to the *follow-me applications* approach in the Sentient Computing project [4] and the linkage shown in Fig. 3c corresponds to services on location-aware portable devices in the Cooltown [6] and NEXUS [5] projects.

## 5 Implementation

The framework presented in this paper was implemented using 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.

### 5.1 Management of locating systems

The current implementation of our framework supports five commercial RFID systems: the RF Code's Spider, Alien Technology's 915 MHz RFID-tag, Philips's I-Code, and Hitachi's mu-chip. The first system provides active RF-tags. Each tag has a unique identifier that periodically emits an RF-beacon (every second) that conveys an identifier via a 305 MHz radio pulse. The system allows us to explicitly control the omnidirectional range of each RF reader to read tags within a range of 1 to 20 m. It can generate enter or leave events when it detects the presence or absence of tags in the range of an RFID reader. As the system's readers have their own batteries, they are portable. The second system provides passive RFID-tags and its readers periodically scan for present tags within a range of 3 m by sending a short 915 MHz-RF pulse waiting for answers. The third system provides passive RFID-tags based on a 13.56 MHz-RF pulse and can scan for present tags within a range of 30 cm. The fourth system provides passive 2.45 GHz-RFID tags and can scan for the presence of tags within a range of 20 cm. The framework converts the resulting list of present tags to enter and leave event notifications by calculating the differences between consecutive scan results. Although there are many differences between the four RFID systems, the framework abstracts these differences away.

### 5.2 Performance evaluation

Although the current implementation of the framework was not built for performance, we measured the cost of migrating a 3 KB agent (zip-compressed) from a source to the destination host that was recommended by the LIS. This experiment was conducted with two LISs and two agent hosts, each of which was running on one of four computers (Pentium III- 1GHz with Windows XP and JDK 1.4). These were directly connected via an IEEE802.11g 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 350 ms, and the cost of agent migration between two hosts over a TCP connection was 39 ms. The latency included the cost of the following processes: UDP-multicasting of the tags' identifiers from the LIS to the source host, TCP-transmission of the agent's requirements from the source host to the LIS, TCP-transmission of a candidate destination from the LIS to the source host, marshaling of the agent, 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 where people are walking.

### 5.3 Security and privacy

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

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

## 6 Applications

This section presents several typical location-based and personalized services that were developed through the framework. Note that these services can be executed at the

[4] This problem is beyond the scope of this paper. Since the framework delegates the execution and migration of agents to its underlying mobile agent platforms, it cannot protect agents from malicious hosts.

same time, since the framework itself is independent of all application-specific services and each service is implemented within mobile agents.

### 6.1 Location-bound universal remote controller

The first example corresponds to Fig. 3a and allows us to use a PDA to remotely control nearby electric lights in a room. Each light was equipped with a tag and was within the range covered by an RFID reader in the room. We controlled power outlets for lights through a commercial protocol called X10. In both approaches described here, the lights were controlled by switching their power sources on or off according to the X10 protocol. Place-bound controller agents, which can communicate with X10-base servers to switch lights on or off, are attached to locations with room lights in this system. Each user has a tagged PDA, which supports the agent host with WindowsCE and a wireless LAN interface (IEEE 802.11b). When a user with a PDA visits a cell that contains a light, the framework moves a controller agent to the agent host of the visiting PDA. The agent, now running on the PDA, displays a graphical user interface to control the light. When the user leaves that location, the agent automatically closes its user interface and returns to its home host. The latency of deployment of the agent at the PDA after the LIS detected the presence of the PDA's tag, was about 3.5 s, where the greater part of latency was in loading the agent to the PDA because the PDA does not have a powerful processor. In the current implementation, the agent is just a collection of Java programs that defined the application logic to display its own GUI on the PDA's screen and the protocol to communicate with the servers, but did not involve any location-based behaviors. This means that we can reuse plain Java-based GUI programs to control lights as location-based service providers with minor modifications. In fact, the size of the agent program was about 8 KB, which is 15% larger than Java-based X10-controller program that offers the same user interface as shown in Fig. 4.

### 6.2 Mobile personal assistance

The second example corresponds to Fig. 3b and offers a user assistant agent that follows the user and maintains profile information about him/her inside itself, so that he/she can always assist the agent in his/her personalized form from any location. Suppose that a user has a 915 MHz-RFID tag and is moving in front of a restaurant, which offers an RFID reader and an agent host with a touch screen. When the tagged user enters inside the coverage area of the reader, the framework enables his/her assistant agents to move to the agent host near his/her current location. After arriving at the host, the agent accesses a database provided by the restaurant to obtain a menu from the restaurant.[5] It then selects appropriate meal candidates from the menu according to his/her profile information, such as favorite foods and recent experiences, stored inside it. It next displays only the list of selected meals on the screen of its current agent host in his/her personalized form. Figure 5 shows a user's assistant agent running on the agent host of the restaurant seamlessly embedding a list of pictures, names, and prices of selected meal candidates with buttons for ordering them through its graphical user interface. Since a mobile agent is a program entity, we can easily define a more intelligent assistant agent. In our early implementation, the cost for the agent to display the selected menu on the screen of a PC (Pentium-III 1-GHz) after the user stopped is within 2 s, where the cost depended on the complexity of the agent. We could easily develop an agent by adding a tiny database to store its user profile and callback methods in response to life-cycle events to a Java applet for selecting and displaying food menus. Also, the agent program is still simple because it does not have to control sensing devices or identify the user.

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

### 6.3 User navigation system

We developed a user navigation system that assists visitors to a building. Several researchers have reported on other similar systems [3, 5]. RF-tags in our system are distributed to several places within the building, such as the ceilings, floors, and walls. As we can see from Fig. 3c, each visitor is carrying a wireless-LAN enabled tablet PC, which is equipped with an RF-tag reader to detect tags, that has access to an LIS and an agent host. The system initially deploys place-bound agents to invisible computers within the building. When a tagged position is located by a cell of the moving RF-tag reader, the LIS running on the visitor's tablet PC detects the presence of the tag. The LIS detects the

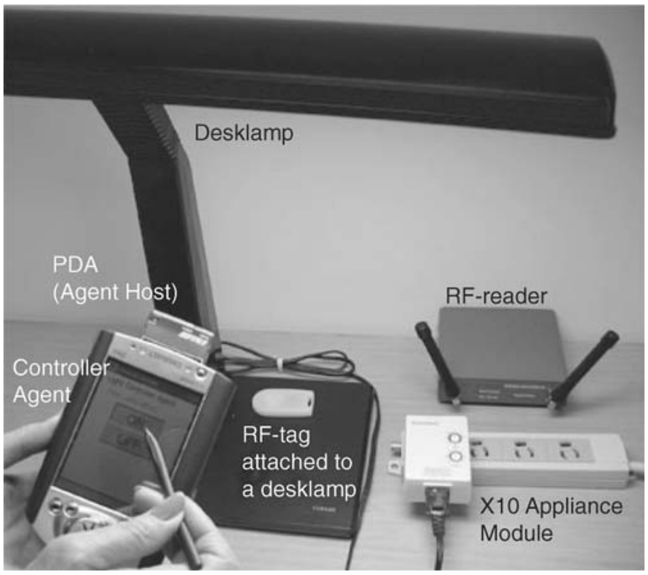

**Fig. 4** Controlling desk lamp from PDA

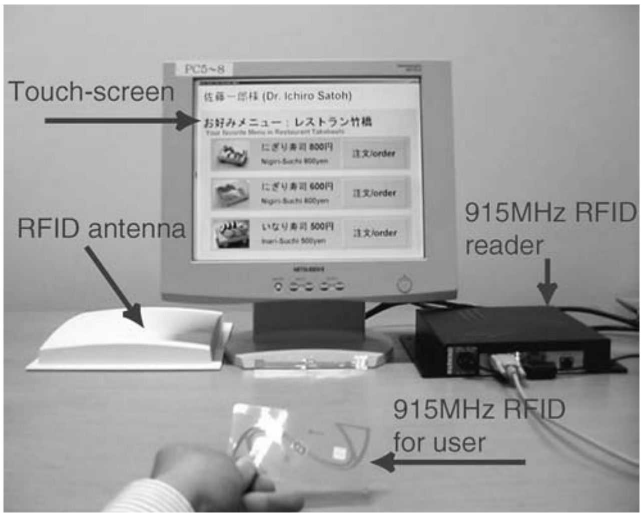

**Fig. 5** Screenshot of follow-me user assistant agent for selecting its user's favorite, i.e., sushi from the menu database of a restaurant that the user is in front of

place-bound agent that is tied to the tag. It then instructs the agent to migrate to its agent host and to provide the agent's location-dependent services at the host. The system enables more than one agent tied to a place to move to the tablet PC. The agents then return to their home computers, and other agents, which are tied to another place, may move to the tablet PC. Figure 6 shows a place-bound agent displaying a map of its surrounding area on the screen of a tablet PC.

### 6.4 Personal server with location sensor

The fourth system corresponds to Fig. 3d. As the personal server proposed by Want [20], it provides a handheld file-sharing server that has no integral user interface but does include secondary storage, a wireless LAN network, and a small 13.56 MHz-RFID reader. RFID tags are located near stationary agent hosts with touch-screens. When a user carries the handheld server near a tagged host, the RFID reader acquires the presence of the tag attached to the host and then the LIS running on the handheld server migrates the agent that is bound to the user to the host. The agent then establishes a TCP connection to the server through a wireless LAN network. It then gathers data from the server and displays this on the screen of the current host. Figure 7 shows that an image viewer agent bound to a user accessing image files from the handheld file server and displaying them on the screen of the agent host near the user's current location. The agent program itself could only provide application logic, which communicates with a server and display the information gathered from the server, independent of any computing devices and locations, because it was always deployed at a suitable computing device by the framework.

## 7 Related work

Much work has been done on location-aware and personalized information systems and existing services can be classified into two types.

The first is to make computing devices move with the user. It often assumes that such devices are attached to positioning systems, such as Global Positioning System (GPS) receivers. For example, HP's Cooltown project [6] is an infrastructure for bridging people, places, and things in the physical world with web resources that are used to store information about them. It allow users to access resources via browsers running on handheld computing devices. All the services available in the Cooltown system are constrained by limitations with web browsers and HTTP. Stuttgart University's NEXUS project [5] provides a platform that supports location-aware applications for mobile users with handheld devices, like the Cooltown project. Unlike our approach, however, both projects aim at providing location-dependent services in outdoor environments and are not suitable for supporting mobile users from stationary computers distributed in a smart environment.

The second approach assumes that a space has been equipped with tracking systems that establish the location of physical entities, including people and objects, within it so that application-specific services can be provided at appropriate computers. Cambridge University's Sentient Computing project [4] provides a location-aware platform using infrared-based or ultrasonic-based locating systems [19] in a building. Using the VNC system [11], the platform can track the movement of a tagged entity, such as individuals and things, so that the graphical user interfaces for the user's

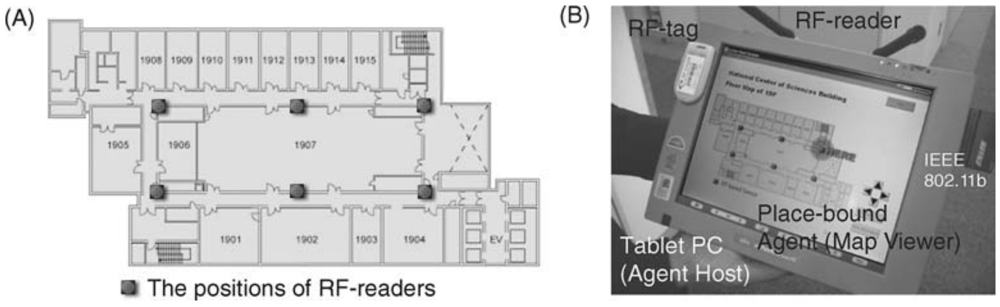

**Fig. 6** Positions of RF-tags in floor (**A**) and screen-shot of map-viewer agent running on table PC (**B**)

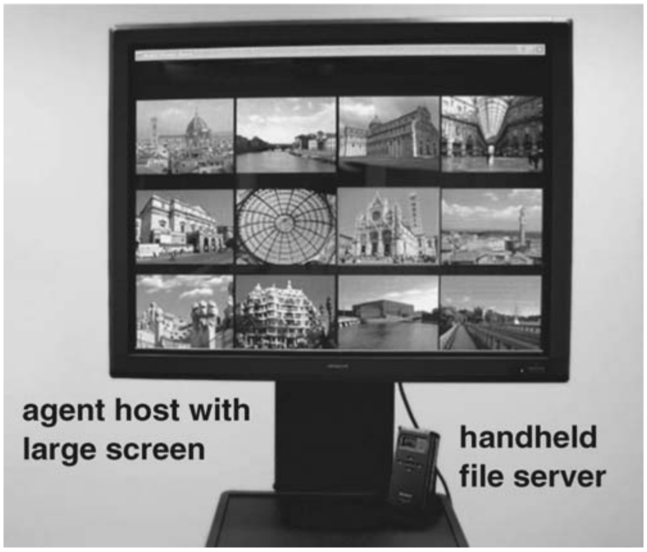

**Fig. 7** Agent host with large-screen and handheld file server

applications follow him/her while he/she is moving around. Since the applications must be executed in remote servers, the platform may have non-negligible interactive latency between the servers and hosts the user accesses locally. A CORBA-based middleware, called LocARE, has been proposed [10] that can move CORBA objects to hosts according to the location of tagged objects. Although the middleware provides similar functionality to that of our framework, its management is centralized and it is difficult to dynamically reconfigure the platform when sensors are added to or removed from the environment.

Microsoft's EasyLiving project [2] provides context-aware spaces, with a particular focus on the home and office. A computer-vision approach is used to track users within the spaces. Both the projects assume that locating sensors have initially been allocated in a room, and it is difficult to dynamically configure the platform when sensors are added to or removed from the environment, whereas our framework permits sensors to be mobile and scatteredly throughout the space. MIT's Project Oxygen Alliance has tried to introduce intelligent spaces into people's lives that are as abundant and natural to use as oxygen in the air, by incorporating several perceptual devices, including location systems. It has provided agent-based infrastructures to construct and manage location-aware services in such spaces [9].

The goal of these infrastructures based on the second approach has been to offer suitable services at suitable locations within the space based on the contextual information within the environment emanating from users, but they have not been able to dynamically deploy service-provider services at suitable computers in the space, as we have done.

Several researchers have explored location-sensitive servers like our LIS. Their location models can be classified into two types: spatial models based on concrete geographical coordinates of objects and spatial models based on 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 [8] 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 similar to the geographical containment model. However, it is unique 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 both be mobile in and to be dynamically added to or removed from a space.

ETH has developed an event-based architecture for managing RF-tags [12]. Like our framework, the architecture can link physical objects with software entities, called virtual counterparts. However, the goal of the architecture is to develop software frameworks that ease the development of particular applications rather than a general framework for supporting various applications. However, it does not support moving computing devices and sensors. Moreover, since the architecture cannot migrate its software entities among ubiquitous computing devices, it cannot effectively support moving objects in the physical world, unlike our framework.

We presented an early prototype of the present framework in previous papers [15], which was just an infrastructure for allowing Java-based agents to follow moving users through locating systems that did not encapsulate application-specific tasks into mobile agents, unlike the framework presented in this paper. We presented another prototype system of the present framework in another previous paper [16]. Since the previous system did not support the mobility of sensors, all the four linkages shown in Sect. 4 were not available in the the previous infrastructure, whereas the framework presented in this paper was designed to be based on RFID-based sensors and therefore permitted the mobility of sensors as well as physical entities, such as people, objects, and computing devices. We presented an extension to the framework that had the ability of managing various location sensors other than RFID-based sensors in another conference paper [18], but this was aimed at integrating the first and second approaches in existing location-aware systems discussed at the beginning of this section using positioning sensors, e.g., Global Positioning System (GPS). The framework presented in this paper, on the other hand, focused on RFID-based sensing systems, because RFID technologies are expected to be widely used in product distribution and inventory management and tags will be placed on many low-cost items, including cans and books in the near future.

## 8 Conclusion

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

Finally, we would like to point out further issues that need to be resolved. Since the framework presented in this paper is general-purpose, in future work we need to apply it to specific applications as well as the three applications presented in this paper. The location model for the framework was designed for operating real location sensing systems in ubiquitous computing environments. We plan to design a more elegant and flexible world model to represent the locations of people, things, and places in the real world by incorporating existing spatial database technologies. We have developed an approach to testing context-aware applications on mobile computers [17]. We are interested in developing a methodology that would test applications based on the framework.

## Appendix: Service provider programs

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

```
class TaggedAgent extends Agent implements
Serializable {
  void go(URL url) throws NoSuchHostException
  {...}
  void duplicate() throws IllegalAccessException
  {...}
  void destroy() {...}
  void setTagIdentifier(TagIdentifier tid) {...}
  void setAgentProfile(AgentProfile apf) {...}
  URL getCurrentHost() {...}
  boolean isConformableHost(HostProfile hfs)
  {...}
  CellProfile getCellProfile(CellIdentifier cid)
    throws NoSuchCellException {...}
  ....
}
```

Let us explain some of the methods defined in the `TaggedAgent` class. An agent executes the `go(URL url)` method to move to the destination host specified as the `url` by its runtime system. The `duplicate()` method creates a copy of the agent, including its code and instance variables. The `setTagIdentifier` method ties the agent to the identity of the tag specified as `tid`. Each agent can specify a requirement that its destination hosts must satisfy by invoking the `setAgentProfile()` method, with the requirement specified as `apf`. The class has a service method named `isConformableHost()`, which the agent uses to decide whether or not the capabilities of the agent hosts specified as an instance of the `HostProfile` class satisfy the requirements of the agent. Also, the `getCellProfile()` method allows an agent to investigate the measurable range and types of RFID readers specified as `cid`.[6]

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

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

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

## References

1. Auto-ID center: http://www.autoidcenter.org/main.asp
2. Brumitt, B.L., Meyers, B., Krumm, J., Kern, A., Shafer, S.: EasyLiving: technologies for intelligent environments, In: Proceedings of the International Symposium on Handheld and Ubiquitous Computing, pp. 12–27 (2000)
3. Cheverst, K., Davis, N., Mitchell, K., Friday, A.: Experiences of developing and deploying a context-aware tourist guide: The guide project. In: Proceedings of the Conference on Mobile Computing and Networking (MOBICOM'2000), pp. 20–31, ACM press, New York (2000)
4. Harter, A., Hopper, A., Steggeles, P., Ward, A., Webster, P.: The anatomy of a context-aware application. In: Proceedings of the Conference on Mobile Computing and Networking (MOBICOM'99), pp. 59–68, ACM Press, New York (1999)
5. 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 the Conference on Mobile Computing and Networking (MOBICOM'99), pp. 249–255, ACM press, New York (1999)
6. Kindberg, T., et al.: People, places, things: web presence for the real world, Technical Report HPL-2000-16, Internet and mobile systems laboratory, HP Laboratories (2000)

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

7. Lange, B.D., Oshima, M.: Programming and deploying java mobile agents with aglets, Addison-Wesley, MA (1998)
8. Leonhardt, U., Magee, J.: Towards a general location service for mobile environments. In: Proceedings of the IEEE Workshop on Services in Distributed and Networked Environments, pp. 43–50, IEEE Computer Society Los Alamitos, CA, USA (1996)
9. Lin, J., Laddaga, R., Naito, H.: Personal location agent for communicating entities (PLACE). In: Proceedings of mobile HCI'02, LNCS, vol. 2411, pp. 45–59, Springer, Berlin Heidelberg New York (2002)
10. Lopez de Ipina, D., Lo, S.: LocALE: a location-aware lifecycle environment for ubiquitous computing. In: Proceedings of the Conference on Information Networking (ICOIN-15), IEEE Computer Society (2001)
11. Richardson, T., Stafford-Fraser, Q., Wood, K., Hopper, A.: Virtual network computing. Proc. IEEE Int. Comput. **2**(1) (2003)
12. Romer, K., Schoch, T., Mattern, F., Dubendorfer, T.: Smart identification frameworks for ubiquitous computing applications. In: Proceedings of the IEEE International Conference on Pervasive Computing and Communications (PerCom'03), pp. 253–262, IEEE Computer Society (2003)
13. Satoh, I.: MobileSpaces: a framework for building adaptive distributed applications using a hierarchical mobile agent system. In: Proceedings of the International Conference on Distributed Computing Systems (ICDCS'2000), pp. 161–168, IEEE Computer Society (2000)
14. Satoh, I.: MobiDoc: a framework for building mobile compound documents from hierarchical mobile agents. In: Proceedings of Symposium on Agent Systems and Applications/Aymposium on Mobile Agents (ASA/MA'2000), Lecture notes in computer science (LNCS), vol. 1882, pp. 113–125, Springer, Berlin Heidelberg New York (2000)
15. Satoh, I.: Physical mobility and logical mobility in ubiquitous computing environments. In: Proceedings of International Conference on Mobile Agents (MA'02), Lecture Notes in Computer Science (LNCS), vol. 2535, pp. 186–202 (2002)
16. Satoh, I.: Location-based services in ubiquitous computing environments. In: Proceedings of the International Conference on Service Oriented Computing (ICSOC'2004), Lecture Notes in Computer Science (LNCS), vol. 2910, pp. 527–542, Springer, Berlin Heidelberg New York (2003)
17. Satoh, I.: A testing framework for mobile computing software. Proc. IEEE Trans. Software Eng. **29**(12), 1112–1121 (2003)
18. Satoh, I.: Linking phyical worlds to logical worlds with mobile agents. In: Proceedings of International Conference on Mobile Data Management (MDM 2004), pp. 332–343, IEEE Computer Society (2004)
19. Want, R., Hopper, A., Falcao, A., Gibbons, J. (1992) The active badge location system. In: Proceedings of the ACM Transactions on Information Systems, vol. 10, no. 1, ACM press, New York pp 91–102 (1992)
20. Want, R.: The personal server—changing the way we think about ubiquitous computing. In: Proceedings of the 4th Interenational Conference on Ubiquitous Computing (Ubicomp 2002), LNCS 2498, pp. 194–290, Springer, Berlin Heidelberg New York (2002)
21. Weiser, M.: The computer for the 21st century, Scientific American, pp. 94–104 (1991)
22. World Wide Web Consortium (W3C): Composite capability/preference profiles (CC/PP), http://www.w3.org/TR/NOTE-CCPP (1999)