Linking Physical Worlds to Logical Worlds with Mobile Agents

Ichiro Satoh*
National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan

Abstract

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

1 Introduction

A goal of ubiquitous computing is to bridge the gap between the physical world and cyberspace. Recent advances in perceptual technologies enable computing devices to detect their surroundings. For example, mobile and ubiquitous computing devices are often equipped with sensors, such as Global Positioning System (GPS) receivers, Radio Frequency IDentification (RFID) readers, and computer-vision cameras, in addition to wireless or wired network interfaces. These sensors have made it possible to detect and track the presence and location of people, computers, and practically any other object we want to monitor. Positioning and tracking systems are likely to become even more ubiquitous in the future. Location-awareness is becoming an essential feature of software applications for ubiquitous and mobile computing devices.

*E-mail: ichiro@nii.ac.jp

Several researchers have explored such location-aware services. Existing services can be classified into two approaches. The first concern is to make the computing devices that move with the user. It often assumes that such devices are attached to positioning systems, such as GPS receivers, which enable them to determine their own locations. For example, in HP's Cooltown project [8], mobile computing devices such as PDAs and smart phones are attached to GPSs to provide location-awareness for web-based applications running on the devices. The second approach assumes that a space is equipped with tracking systems which establish the location of physical entities, including people and objects, within the space so that application-specific services can be provided to appropriate computers. A typical example of this is the so-called follow-me application, which was a study by Cambridge University's Sentient Computing project [5], to support ubiquitous and personalized services on computers located near users.

The two approaches exist at posed as polar opposites, although their final goals coincide. This paper presents an infrastructure for integrating the two approaches that is designed to mitigate the disadvantages of one by juxtaposing these with the advantages of the other. In other words, the infrastructure does not distinguish between mobile and stationary computing devices. It can permit tracking sensors to be moved with the user and dynamically added to and removed from a space it does this, because it dynamically creates a world model when detecting the appearance and movement of sensors and physical entities including people, objects, and computing devices. Moreover, it is unique among other existing location-aware systems in that it uses mobile agent technology. It enables these agents to be spatially bound to people, places, and objects, which the agents then support and annotate. By using tracking systems, the infrastructure dynamically deploys such agents to stationary and mobile computing devices that are near the entities and places, even when the locations of the entities change. Using mobile agents makes the infrastructure independent of any applications, as application-specific services are implemented within mobile agents instead of the infrastructure.

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

2 Approach

The framework presented in this paper aims to enhance the capabilities of users, particularly mobile users, things, including computing devices and non-electronic objects, and places, such as rooms, buildings and cities. All these elements must have suitable computational functionalities that enable their support and annotation.

2.1 Location-sensing Systems

There have been a variety of location-sensing systems. They can be classified into two types: tracking and positioning systems. The former, including RFID, measures the location of other located objects. The latter, including GPS, measures its own location. Tracking sensors can be embedded in the environment and positioning sensors can be carried with portable computing devices. There are two different ways to locate objects: geometric location and symbolic location. The former represents the locations of objects by reporting their relative or absolute coordinates in a frame of reference. The latter determines the location of objects by identifying the names of the spatial regions which contain them.

This framework must 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 of the underlying tracking and positioning systems. Therefore, the model is based on the 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. The current implementation supports location-sensing systems, in which spatial regions can be determined within a few square feet, to distinguish between one or more portions of a room or building. It assumes that the underlying sensing systems can detect not only the locations of entities and places we want to monitor but also the locations of computing devices that execute and migrate mobile agents.

2.2 Application-specific Services

Suitable services should be operated on suitable computing devices in the sense that the services are both wanted according to the location of users and their associate 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 suitable location for a wanted service to be provided, the device may not be available because of a lack of software or capabilities, such as input or output facilities, for executing 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 this limitation, 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 also can 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 the services can be

[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., mobile agent.

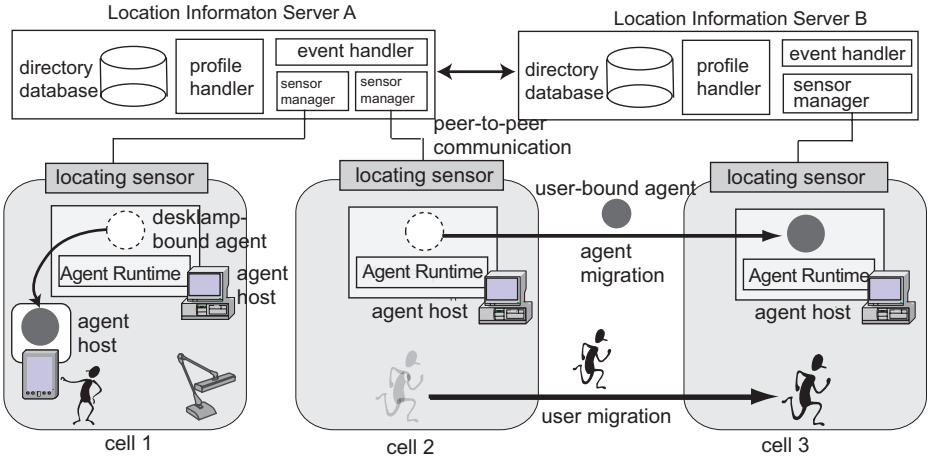

Figure 1. Architecture of infrastructure.

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 the client-side and service side computers.

3 Design and Implementation

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

3.1 Location Information Servers

Each LIS manages more than one sensor to detect the locations of entities and agent hosts and provide mobile agents with up-to-date information on destinations. Its sensors detect the movement of the physical entities or places the agents are bound to. Figure 2 shows the basic structure of a LIS. However, since not only physical entities but also agent hosts and sensors may be movable, it is almost impossible to maintain a geographical model of the whole system. To solve this problem, the infrastructure provides a demand-driven mechanism to discover the agents and agent hosts required by mobile ad-hoc network approaches. LISs are individually connected to other servers with a peer-to-peer connection to exchange information though TCP sessions. They can also send control-information through multicast communications.[2]

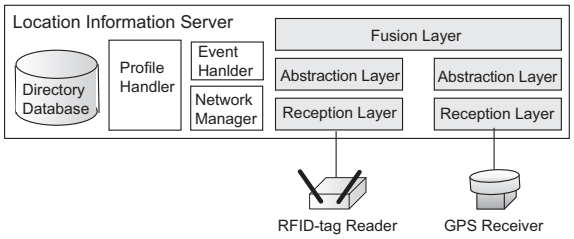

Figure 2. Location information server

Management of Location Sensors:

Each LIS manages more than one sensor and agent hosts, and maintains up-to-date information on the identities of those that are within the zone of coverage by means of its sensors. To hide the differences between the underlying location-sensors, each LIS provides an abstract three-layer stack. This can be mapped to a number of architectures to provide the acquisition function as in the acquisition stack [11].

- The *Reception* layer is responsible for extracting the data from the sensors, as sensors generally tend to be proprietary or vendor-specific. For example, some sensors can be retrieved at any time through synchronous

[2]The current implementation relies on multicast communications because the LISs that detect the presence of the same located-object are located within a localized space smaller than the reachable domain of the multicast packets

queries and other sensors can issue results continuously or periodically. The layer polls sensors or receives events issued from sensors.

- The *Abstraction* layer receives low-level data about the locations of entities from sensors and then transforms the data in a symbolic model. For example, the current implementation maps geometric locations measured by sensors, e.g., GPS and wireless and cellular network, into specified regions, e.g., one or more portions of a room or building. When an RFID reader detects the presence of a tagged entity, the location of the entity is represented as the identifier of the reader. We call each sensor's coverage and each region a *cell*, as location models studied by several other researchers [10].
- The *Fusion* layer correlates the sightings belonging to the same located-object from different sensors. This infrastructure allows sensors to be mobile and throughout a space. When one or more cells overlap geographically, an entity may be in multiple cells at the same time and each of the LISs that manage the cells sends update information to agents bound to the entity.

The above layers are implemented as a collection of mobile agents as configurable middleware for sensor networks [25]. They can be dynamically changed according to the specifications of sensors and the requirements of applications.

Mechanism for Agent Discovery:

Each LIS is responsible for discovering the mobile agents bound to entities within its cells. It maintains a database in which it stores information about each of the agent hosts and each of the mobile agents attached to an entity or place. When an LIS detects a new entity, the LIS multicasts a query containing the name of the new entity, i.e. the identity of the tag attached to the entity, and its own network address to all the agent hosts in its current sub-network. It then waits for reply messages from the agent hosts. There are two possible cases: the tag may be attached to an agent host or it may be attached to a person, place, or thing other than an agent host.

- In the first case the newly arrived agent host will send its network address and device profile to the LIS. The profile describes the capabilities of the agent host, e.g. its input devices and screen size. After receiving this reply, the LIS stores the profile in its database and forwards it to all agent hosts within the cell.
- In the second case agent hosts with 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 can be visited and used.

The LIS then stores the requirements of the agents in its database and moves the agents to appropriate agent hosts in the way 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 an entity, or an RFID tag attached with the entity, is detected in a cell, each LIS multicasts a message with the identifier of the entity and the identifier of the cell to all agent hosts in its current sub-network. Figure 3 shows a sequence for migrating an agent to a proper host when an LIS detects the presence of a new RFID tag.

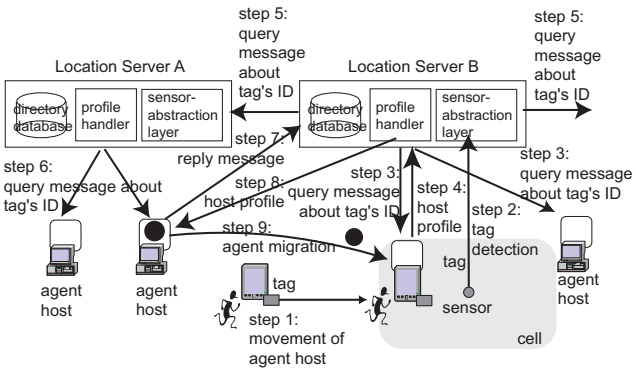

Figure 3. Agent discovery and deployment

Agent Navigation

We will explain how agents navigate to appropriate agent hosts. When an LIS detects the movement of an entity, e.g. a person or object, that is being tracked by a sensor or an RFID tag attached to the entity in a cell, it searches its database for agent hosts that are present in the current cell of the entity or RFID tag. It also selects candidate destinations from the set of agent hosts within the cell, according to the respective capabilities. The infrastructure offers an XML-based language based on CC/PP (composite capability/preference profiles) [28]. The language describes the capabilities of agent hosts and the requirements of mobile agents using a CC/PP notation. For example, a description contains information on the following properties of a computing device: the vendor and model class of the device (i.e, PC, PDA, or phone), its screen size, number of colors, CPU, memory, input devices, secondary storage, and the presence/absence of loudspeakers. The infrastructure allows each agent to specify the preferable capabilities of agent hosts that it may visit, as well as the minimal capabilities using 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 bound to the entities or places. The message specifies the profiles of the agent hosts present in the cell and satisfies the requirements of the agent. Since mobile agents can travel through a network under their own control, LISs cannot always know the current location of the mobile agents. When LISs detect changes in the physical world, they notify each agent of the network address and the capabilities of more than one candidate destination that the agent should visit. However, they never send the agent to the destination, because mobile agents should be treated as autonomous entities. Each agent then selects one host from the list of candidates recommended by the LIS and migrates to that host. Moreover, when the capabilities of such a destination do not satisfy the requirements of an agent, the agent itself should decide whether or not to migrate itself to the destination and adapt itself to the limited capabilities according to its own configuration policy.

3.2 Mobile Agent

The infrastructure encapsulates application-specific services into mobile agents so that it is independent of any applications and can support multiple services. As mentioned in the appendix, each mobile agent is a collection of Java objects and is equipped with the identifier of a locatable entity or the identifier of an RFID tag to which the agent is attached. The agent is a self-contained program and is able to communicate with other agents. An agent 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 [19].

3.3 Agent Host

Each agent host must be located by the location-sensors. For example, an agent host may be equipped with an RFID tag. It has two forms of functionality: one for advertising its capabilities and another for executing and migrating mobile agents. The current implementation assumes that LISs and agent hosts can be directly connected through a wired LAN such as Ethernet, or a wireless LAN, such as IEEE802.11b. When a host receives a query message with the identifier of a newly arrived 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; (ii) if one of the agents running on it is tied to the tag, it returns its network address and the agent requirements; and (iii) if neither of the above cases applies, it ignores the message.

The current implementation of this infrastructure is based on a Java-based mobile agent system called MobileSpaces [18].[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 agents inside it and maintains the life-cycle state of each. When the life-cycle state of an agent changes, for example when it is created, terminates, or migrates to another host, the runtime system issues specific events to the agent. This is because the agent may have to acquire or release various resources, such as files, windows, or sockets, which it had previously captured. When notification of 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.

4 Four Linkages Physical Worlds to Logical Worlds

The framework does not have to distinguish between mobile and stationary computing devices and 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 4 illustrates the four linkages when entities and agent hosts are attached to RFID tags.

- Figure 4 (a) 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 location-dependent services of the place.
- Figure 4 (b) 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 4 (c) 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

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

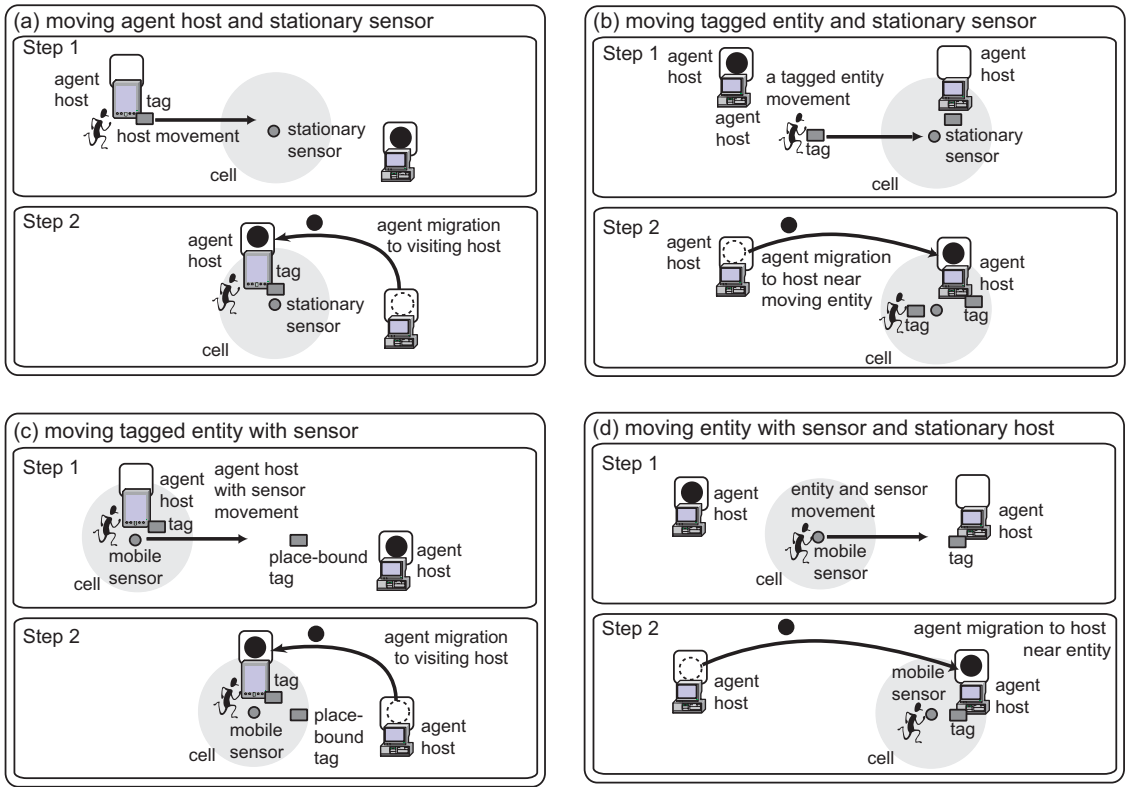

Figure 4. Four linkages between physical and logical worlds

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 4 (d) 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 source of location information, e.g., GPS, local wireless networks, and cellular networks. Existing location-aware systems can only support each of the above linkages, whereas our infrastructure does not have to distinguish between them and can synthesize them seamlessly. For example, the linkage shown in Figure 4 (a) supports *person tracking display* approach in the EasyLiving project [1], the linkage shown in Figure 4 (b) supports *Follow-me applications* approach in the Sentient Computing project [5] and the linkage shown in Figure 4 (c) supports services on location-aware portable devices studied in the Cooltown [8] and NEXUS [6] projects.

4.1 Current Status

The infrastructure 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.

4.2 Management of Locating Systems

The current implementation of our infrastructure supports five commercial RFID system: RF Code's Spider system, Alien Technology's 915MHz RFID-tag system, Philips I-Code system, and Hitachi's mu-chip system. The first system provides active RF-tags. 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 meters. It can generate enter or leave events when it detects the presence or absence of tags in the range of an RFID reader. The system's readers can have their own batteries, so that they can be portable. The second system provides passive RFID-tags and its readers periodically scan

for present tags within a range of 3 meters by sending a short 915 MHz-RF pulse and waiting for answers. 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 centimeters. The fourth system provides passive 2.45 GHz-RFID tags and can scan the presence of tags within a range of 20 centimeters. The infrastructure 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 infrastructure abstracts these differences away. The current implementation also supports a commercial GPS receiver system. The infrastructure maps geometric locations measured by a GPS into specified regions.

Performance Evaluation:

Although the current implementation of the infrastructure 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 2000 and JDK 1.4). These 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 msec, and the cost of agent migration between two hosts over a TCP connection was 48 msec. The latency includes the cost of the following processes: UDP-multicasting of the 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 for people walking.

Security and Privacy

Security is essential in mobile agent computing. The infrastructure can be built on many Java-based mobile agent systems with the Java virtual machine. Therefore, it can directly use the security mechanism of the underlying mobile agent system. This can explicitly restrict agents to only access specified resources in order to protect hosts from malicious agents. To protect against the arrival of malicious agents from agent hosts, the MobileSpaces system supports a Kerberos-based authentication mechanism for agent migration [24]. It authenticates users without exposing their passwords on the network, and generates secret encryption keys that can selectively be shared by mutually suspicious parties.

The infrastructure 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 movement. 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 transform agents into an encrypted form 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.

5 Applications

This section presents several typical location-based services developed using this infrastructure. Note that all the services presented in this section can coexist at the same time since the infrastructure itself is independent of any application-specific services and each service is implemented within mobile agents.

5.1 Location-aware Universal Remote Controller

The first system corresponds to Figure 4 (a). It allows us to use a PDA to remotely control nearby lights. In this system place-bound controller agents, which can communicate with X10-base servers to switch lights on or off, are attached to places with room lights. The PDA and desktop lamp are attached with Spider RFID tags. Each user has an RFID-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 an RFID-tagged desktop lamp, the infrastructure 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 lamp. When the user leaves that location, the agent automatically closes its user interface and returns to its home host.

5.2 Personalized Services in Anywhere

The second system corresponds to Figure 4 (b). It tracks the current location of a user by using an Alien Technology's 915 MHz-RFID system and allow themselves to access the user's applications at the nearest computer as they move around within the building. We present two *follow-me* agents. The first agent provides a desktop *teleporting* system, like a *follow-me* application [5]. Unlike previous

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

Figure 5. Controlling desk lamp from PDA

studies of such applications, our infrastructure cannot only migrate the user interfaces of applications but also the applications themselves to appropriate computers in the cell that contains the tag of the user. In our previous paper [19] we developed a mobile window manager, which is a mobile agent that can carry its desktop applications as a whole to another computer and control the size, position, and overlap of the windows of the applications. Using the infrastructure presented in this paper, the window manager and desktop applications can be automatically moved to and then executed on the computer that is in the current cell of the user and has the resources required by the applications. Figure 6 shows the migration of desktop applications between two agent hosts.

The second agent is a user assistant agent that follows its user and maintains profile information about the user inside itself, so that the user can always assist the agent in a personalized manner anywhere. Suppose that a user has a 915 MHz-RFID tag and is moving in front of a restaurant that offers an RFID reader and agent host with a touch-screen. When the tagged user enters the coverage area of the reader, the infrastructure enables the assistant agents to be automatically moved to the agent host near the user's current location. After arriving at the host, the agent accesses a database provided in the restaurant to get a menu from the restaurant. [5] It then selects appropriate meal candidates from the menu according to the user's profile information, such as favorite foods and recent experiences. Next, it displays only the list of the selected meals on the screen of its current agent host in a personalized manner. Figure 7 shows that a user's assistant agent runs on the agent host of the restaurant and seamlessly embeds the list of pictures, names, and prices of the selected meal candidates with buttons for ordering them into its graphical user interface. Since a mobile agent

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

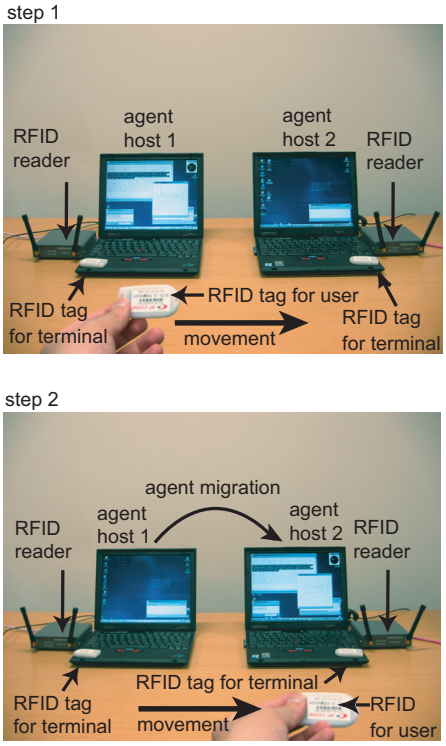

Figure 6. Follow-me desktop applications between two computers

is a program entity, we can easily define a more intelligent assistant agent.

5.3 Location-based Services on Mobile Computing Devices

The third system corresponds to Figure 4 (c). It provides a user navigation system that assists visitors in a building. Several researchers have reported on other similar systems [2, 6]. In our system tags are distributed to several places within the building, such as its ceilings, floors, and walls. Each visitor carries a wireless-LAN enabled tablet PC, which is equipped with a GPS receiver to measure its own position or an RFID reader to detect tags, and includes an LIS and an agent host. The system initially deploys place-bound agents to invisible computers within the building. When the GPS receiver finds itself in a specified place or the RFID reader detects the presence of the tag bound to the place, the LIS running on the visitor's tablet PC detects the place-bound agent that is bound to the place. It then instructs the agent to migrate to its agent host and provide the agent's location-dependent services at the host. The system enables more than one agent tied to a place to move to the table PC; the agent then returns to its home computer and

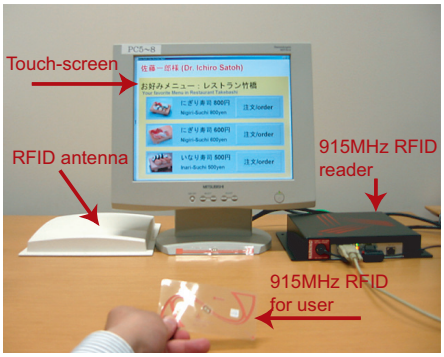

Figure 7. Screenshot of follow-me user assistant agent selecting user's favorite foods from database

other agents, which are tied to another place, moves to the tablet PC. Figure 8 shows a place-bound agent to display a map of its surrounding area on the screen of a tablet PC.

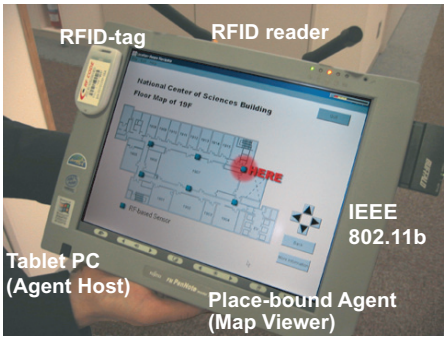

Figure 8. (A) location of RF-tags in floor (B) and screen-shot of map-viewer agent running on table PC

5.4 Integrating Handheld Computers with Stationary User Interface Devices

The fourth system corresponds to Figure 4 (d). As the personal server proposed by Want [27], 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 with 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. Next it gathers data from the server and then displays the data on the screen of the current host. Figure 9 shows that an image viewer agent bound to a user accesses image files from the handheld file server and then display the files on the screen of the agent host near the user's current location.

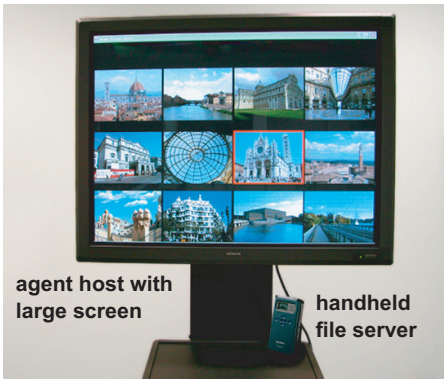

Figure 9. Agent host with large-screen and handheld file server

6 Related Work

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

We compared our approach with several projects that support mobile users in a ubiquitous computing environment. Research on smart spaces and intelligent environments has become popular at many universities and corporate research facilities. Cambridge University's Sentient Computing project [5] provides a platform for location-aware applications using infrared-based or ultrasonic-based locating systems in a building. Using the VNC system [15] the platform can track the movement of tagged entities, such as individuals and things, so that the graphical user interfaces of the user's applications follow them while they are moving around. Although the platform provides similar functionality to of our approach, its management is centralized and thus it is difficult to dynamically reconfigure the platform when sensors are added to or removed from the environment. Since the applications must be executed in remote servers, the platform may have non-negligible interactive latency between the servers and the hosts that the user accesses locally. Our approach, however, enables a user's application, 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 in-

teractions between him/her and the applications. Recently, the project provided a CORBA-based middleware system called LocARE [13]. The middleware can move CORBA objects to hosts according to the location of tagged objects, However CORBA objects are not always suitable for implementation on user interface components.

Microsoft's EasyLiving project [1] provides context-aware spaces, with a particular focus on the home and office. It uses mounted sensors, such as stereo cameras, on the room's walls and tracks the locations and identities of people in the room. The system can dynamically aggregate network-enabled input/output devices, such as keyboards and mice, even when they belong to different computers in the space. However, its management is centralized and it does not dynamically migrate software to computers according to the position of 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. Our approach, however, permits sensors to be mobile and scattered throughout the space.

MIT's Project Oxygen Alliance has tried to introduce intelligent spaces that are as abundant and accessible to use as oxygen into people's lives by incorporating several perceptual devices, including location systems. It has provided agent-based infrastructures to construct and manage location-aware services in such spaces [12]. The goal of these infrastructures has been to offer suitable services at suitable locations within the space based on contextual information within the environment and information emanating from users. However, they have not been able to dynamically deploy service-provider services at suitable computers in the space, as we have done.

There have also been several studies on enhancing context-awareness in mobile computing. HP's Cooltown [8] 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 with the web resources that are used to store information about them. The bridges that it forms allow users to access resources stored on the web via a browser using standard HTTP communication. Although user familiarity with web browsers is an advantage in this system, all the services available in the Cooltown system are constrained by the limitations of web browsers and HTTP. Our approach, however, is not limited by a 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 [6], 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 GPS-based positioning sensors and wireless communication. Applications that run on such devices (e.g. user-navigation) maintain a spatial model of the current vicinity of users and gather spatial data from remote servers. Unlike our approach, however, neither Cooltown nor NEXUS can support mobile users through stationary computers distributed in a smart environment.

Several research projects have introduced software mobility as a technology for enabling ubiquitous computers to support various services, which they may have not been initially designed for. The Aura project [3] of CMU and the Gaia project [16] of the University of Illinois at Urbana-Champaign provide infrastructures for binding tasks associated with users, and migrating applications from computer to computer as users move about, like our approach does. Although they share several common design goals with our framework, they focus on the development of contextual services for users rather than the location-aware deployment of services. Kangas [7] developed a location-aware augmented-reality system that enables the migration of virtual objects to mobile computers, but only when the computer was in a particular space, in a similar way to our framework. However, the system is not designed to move such virtual objects to ubiquitous computing devices. The one.world project [4] by the University of Washington provides a middleware infrastructure for ubiquitous computing, but does not provide any location-aware mechanisms for deploying services at computing devices. It assumes a distributed shared memory and builds applications based on the principle of separating data and functionality in applications, where our approach always treats applications as a set of data and functionality to be deployed at various devices that is not initially designed for executing the application. Hive [14] is a distributed agent middleware for building decentralized applications. It can deploy agents at devices in ubiquitous computing environments and organize the devices as groups of agents. Although it can provide contextual information for agents, it does not support any mechanism for monitoring sensors and deploying agents according to changes in the environment, unlike ours.

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 [10] 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 mod-

els. 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. In previous papers [21, 22], we presented an early prototype of the present framework. This approach does not support the mobility of sensors and agent hosts so that the four linkages described in the second section of this paper were not available in the the previous framework unlike the framework presented in this paper.

7 Conclusion

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

Finally, we would like to point out further issues to be resolved. Since the framework presented in this paper is general-purpose, in future work we need to apply it to specific applications as well as the three applications presented in this paper. The location model of the framework was designed for operating real location-sensing systems in ubiquitous computing environments. We plan to design a more elegant and flexible world model for representing the locations of people, things, and places in the real world by incorporating existing spatial database technologies. We have developed an approach to testing context-aware applications on mobile computers [20, 23]. We are interested in developing a methodology that would test applications based on the framework.

References

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

[2] K. Cheverst, N. Davis, K. Mitchell, and A. Friday: Experiences of Developing and Deploying a Context-Aware Tourist Guide: The GUIDE Project, Proceedings of Conference on Mobile Computing and Networking (MOBICOM'2000), pp. 20-31, ACM Press, 2000.

[3] D. Garlan, D. Siewiorek, A. Smailagic, and P. Steenkiste, " Project Aura: Towards Distraction-Free Pervasive Computing, IEEE Pervasive Computing, vol. 1, pp. 22-31, 2002.

[4] R. Grimm, el. al.: Systems Directions for Pervasive Computing, Proceedings of 8th Workshop on Hot Topics in Operating Systems, pp.147-151, May 2001.

[5] A. Harter, A. Hopper, P. Steggeles, A. Ward, and P. Webster: The Anatomy of a Context-Aware Application, Proceedings of Conference on Mobile Computing and Networking (MOBICOM'99), pp. 59-68, ACM Press, 1999.

[6] F. Hohl, U. Kubach, A. Leonhardi, K. Rothermel, and M. Schwehm: Next Century Challenges: Nexus - An Open Global Infrastructure for Spatial-Aware Applications, Proceedings of Conference on Mobile Computing and Networking (MOBICOM'99), pp. 249-255, ACM Press, 1999).

[7] K. Kangas and J. Roning: Using Code Mobility to Create Ubiquitous and Active Augmented Reality in Mobile Computing, Proceedings of Conference on Mobile Computing and Networking (MOBICOM'99), pp. 48-58, ACM Press, 1999.

[8] T. Kindberg, et al: People, Places, Things: Web Presence for the Real World, Technical Report HPL-2000-16, Internet and Mobile Systems Laboratory, HP Laboratories, 2000.

[9] B. D. Lange and M. Oshima: Programming and Deploying Java Mobile Agents with Aglets, Addison-Wesley, 1998.

[10] U. Leonhardt and J. Magee: Towards a General Location Service for Mobile Environments, Proceedings of IEEE Workshop on Services in Distributed and Networked Environments, pp. 43-50, IEEE Computer Society, 1996.

[11] U. Leonhardt and J. Magee: Multi-Sensor Location Tracking, Proceedings of Conference on Mobile Computing and Networking (MOBICOM'98), pp.203-214, ACM Press, 1998.

[12] J. Lin, R. Laddaga, and H. Naito: Personal Location Agent for Communicating Entities (PLACE) Proceedings of Mobile HCI'02, LNCS, Vol. 2411, pp. 45-59, Springer, 2002.

[13] D. Lopez de Ipina and S. Lo: LocALE: a Location-Aware Lifecycle Environment for Ubiquitous Computing, Proceedings of Conference on Information Networking (ICOIN-15), IEEE Computer Society, 2001.

[14] N. Minar, M. Gray, O. Roup, R. Krikorian, and P. Maes: Hive: Distributed agents for networking things, Proceedings of Symposium on Agent Systems and Applications / Symposium on Mobile Agents (ASA/MA'99), IEEE Computer Society, 2000.

[15] T. Richardson, Q, Stafford-Fraser, K. Wood, A. Hopper: Virtual Network Computing, IEEE Internet Computing, Vol. 2, No. 1, 1998.

[16] M. Román, C. K. Hess, R. Cerqueira, A. Ranganat R. H. Campbell, K. Nahrstedt K, Gaia: A Middleware Infrastructure to Enable Active Spaces, IEEE Pervasive Computing, vol. 1, pp.74-82, 2002.

[17] K. Romer and T. Schoch: Infrastructure Concepts for Tag-Based Ubiquitous Computing Applications Workshop on Concepts and Models for Ubiquitous Computing, Ubicomp 2002, September 2002.

[18] I. Satoh: MobileSpaces: A Framework for Building Adaptive Distributed Applications Using a Hierarchical Mobile Agent System, Proceedings of Conference on Distributed Computing Systems (ICDCS'2000), pp. 161-168, IEEE Computer Society, 2000.

[19] I. Satoh: MobiDoc: A Framework for Building Mobile Compound Documents from Hierarchical Mobile Agents, Proceedings of Symposium on Agent Systems and Applications / Symposium on Mobile Agents (ASA/MA'2000), LNCS, Vol. 1882, pp. 113-125, Springer, 2000.

[20] I. Satoh: Flying Emulator: Rapid Building and Testing of Networked Applications for Mobile Computers, Proceedings of Conference on Mobile Agents (MA'01), LNCS, Vol. 2240, pp. 103-118, Springer, 2001.

[21] I. Satoh: Physical Mobility and Logical Mobility in Ubiquitous Computing Environments, Proceedings of Conference on Mobile Agents (MA'02), LNCS, Vol. 2535, pp. 186-202, Springer, 2002.

[22] I. Satoh: Location-based Services in Ubiquitous Computing Environments, to appear in Proceedings of International Conference on Service Oriented Computing (ICSOC'2004), LNCS, Springer, December 2003.

[23] I. Satoh: A Testing Framework for Mobile Computing Software, to appear in IEEE Transactions on Software Engineering, (Accepted) vol.29, 2003.

[24] I. Satoh: Configurable Network Processing for Mobile Agents on the Internet, Cluster Computing, (Accepted) vol. 7, no.1, Kluwer, January 2004.

[25] T. Umezawa I. Satoh, and Y. Anzai, A Mobile Agent-based Framework for Configurable Sensor Networks. Proceedings of International Workshop on Mobile Agents for Telecommunication Applications (MATA'2002), Lecture Notes in Computer Science, Springer, Vol. 2521, pp. 128-140, 2002.

[26] R. Want, A. Hopper, A. Falcao, and J. Gibbons: The Active Badge Location System, ACM Transactions on Information Systems, vol.10, no.1, pp. 91-102 ACM Press, 1992.

[27] R. Want: The Personal Server - Changing the Way We Think about Ubiquitous Computing, Proceedings of 4th International Conference on Ubiquitous Computing (Ubicomp 2002), LNCS 2498, pp. 194-209, Springer, September 2002.

[28] World Wide Web Consortium (W3C): Composite Capability/Preference Profiles (CC/PP), http://www.w3.org/TR/NOTE-CCPP, 1999.

Appendix: Mobile Agent Programs

This section explains the programming interface for our 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 hpf) {...}
  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 have more than one listener object that implements a specific listener interface to hook certain events issued before or after changes in its life-cycle state or the movements of its tag.

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

The above interface specifies the fundamental methods that are invoked by the runtime system when agents are created, destroyed, or 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.

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