Software Agents for Ambient Intelligence*

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

Abstract – *This paper presents a general-purpose framework for ambient intelligences. The framework provides people, places, and things with computational functionalities to assist and annotate them. Using RFID-based location-tracking systems, it can navigate Java-based mobile agents. These agents can offer location-aware and personalized services, to stationary or mobile computers near the locations of the entities and places to which the agents are attached, even when the locations change. The framework enables users to directly access their personalized services from stationary computing devices in the environment or from their portble computing devices, even when they move.*

Keywords: ubiquitous computing, sensor, mobile agent, RFID

1 Introduction

Ambient intelligence enables us to be surrounded by electronic environments that are sensitive and responsive to people. Ambient intelligence technologies are expected to combine concepts of intelligent systems, perceptual technologies, and ubiquitous computing. Perceptual technologies have made it possible to measure and track the locations of people, computers, and practically any other object of interest. For example, indoor location systems, such as RFID (radio frequency identification) tags, detect the locations of physical entities in a building and enable applications to respond to these locations. Location awareness is becoming an essential feature of services targeted at ambient intelligence. Several researchers have explored such location-aware services. Existing services can be classified into two approaches. The first is to have the computing devices move with the user; the devices are attached to positioning systems, which enable them to measure their own locations. The second approach assumes that a space is equipped with tracking systems that establish the locations of physical entities, including people and objects, within it so that application-specific services can be provided at appropriate computers. The two approaches are posed as polar opposites, although their final goals seem to coincide.

This paper presents a framework for integrating the two approaches to reinforce their advantages by using mobile agent technology. The framework enables mobile agents to be spatially bound to people, places, and things that the agents assist and annotate. By using tracking systems, the framework dynamically deploys such agents to stationary and mobile computing devices that are near the locations of the entities and places to which the agents are attached, even when the locations of the entities change. Using mobile agents makes the framework application-independent; application-specific services are implemented within mobile agents instead of the infrastructure and become intellgent and proactive. Therefore, the framework enables various location-aware and personalized services to be constructed.

2 Related Work

Research on ambient intelligence or smart spaces has become common at many universities and corporate research facilities. For example, Cambridge University's Sentient Computing project [2] provides a platform for building location-aware applications by using locating systems. Unlike our framework, the management of the the platform is centralized and cannot dynamically reconfigure itself. The project uses CORBA-based middleware can move CORBA objects to hosts according to the location of tagged objects [7]. However, CORBA objects are not well suited for implementing user interface components and migrating between heterogeneous platforms. Microsoft's EasyLiving project [1] provides context-aware spaces, with a particular focus on home and office. This project can dynamically aggregate network-enabled input/output devices, such as keyboards and mice, even when they belong to different computers in a space. However, its management is centralized, and it does not dynamically migrate software to other computers according to the location of users. Both projects assume that locating sensors have initially been allocated in the room, and that dynamically reconfiguring the platform is difficult when sensors are added to or removed from the environment. In contrasts, our framework permits sensors to be mobile and scattered throughout a space.

Several studies have focused on enhancing context-awareness in mobile computing, for example HP's Cooltown

*0-7803-8566-7/04/$20.00
†E-mail: ichiro@nii.ac.jp

[4] and the NEXUS system [3]. These projects assume that each user has a notebook PC, tablet PC, or PDA, equipped with GPS-based positioning sensors and wireless communication. Applications that run on such devices access resources stored on the web via a browser by using standard HTTP communication.

Although user familiarity with web browsers is an advantage in these systems, the services available are constrained by the limitations of web browsers and HTTP. In contrast, our framework can dynamically deploy mobile agents, which are autonomous programmable entities, to mobile computing devices. Unlike our approach, neither Cooltown nor NEXUS can support mobile users through stationary computers distributed in a smart environment.

While many researchers have explored mobile agent technology, no attempts have been made to integrate the mobility of physical objects with the mobility of agents in a ubiquitous computing setting. We previously presented an early prototype of the present framework [9] that did not support the mobility of sensors and agent hosts, so that the three linkages described in the second section of this paper were not available in that previous version of the framework.

3 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, that have the computational functionalities to assist and annotate them.

3.1 Location-sensing Systems

Our goal is to provide a location-aware system in which spatial regions can be determined within a few square feet, so that one or more portions of a room or building can be distinguished. The current implementation uses RFID tag technology to locate objects. An RFID system uses RF (radio frequency) readers that detect the presence of small RF transmitters, often called *tags*. The framework assumes that physical entities, including people and computing devices, and places are equipped with these tags so that they are automatically locatable. It spatially binds software for information-based services to an RFID tag attached to a person, place, or thing in the physical world.

3.2 Location-based and Personalized Services

An ambient intelligent computing system computing environment consists of many computing devices, which may have only limited resources, such as restricted levels of CPU power and limited amounts of memory. As a result, even if a device is at a location suitable for providing a wanted service, the device may not be able to do so due to a lack of capabilities, such as input or output facilities. To overcome this limitation, the framework introduces mobile agent technology the agents of which only needs to be present at the computer during the time the computer uses the services provided by that agent. Mobile agent technology also has the following advantages.

- Various kinds of infrastructures have been used to construct and manage location-aware services. However, such infrastructures have mostly focused on a particular application, such as user navigation. By separating application-specific services from the infrastructure, our framework provides a general infrastructure for a variety of location-aware services, enabling application-specific services to be implemented within mobile agents.
- Each mobile agent can dynamically be deployed at and locally executed within computers near the position of the user. As a result, the agent can directly interact with the user, whereas RPC-based approaches, on which other approaches are often based, must have network latency between the local computer and remote servers. The agent also can directly access various equipment belonging to that device as long as the security mechanisms of the device permit this.
- After arriving at its destination, a mobile agent can continue working without losing any previous results of work at the source computers, e.g., the content of instance variables in the agent's program. Thus, the technology enables ease of building follow-me applications as proposed as proposed Harter et al. [2].

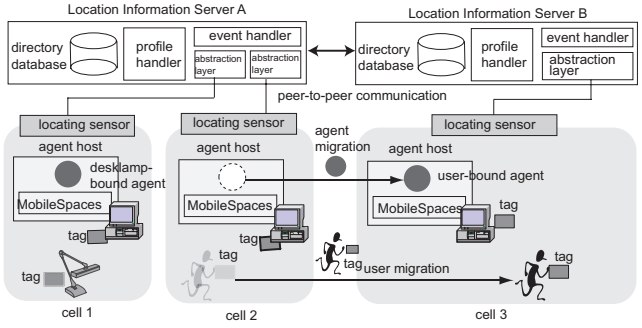

Figure 1: Architecture of the SpatialAgent framework

4 Design and Implementation

The framework consists of three parts: (1) location information servers, called LISs, (2) mobile agents, and (3) agent hosts. The first part provides a layer of indirection between the underlying locating sensing systems and mobile agents. As shown in Fig. 1, each LIS manages more than one sensor and provides the agents with up-to-date information on the identifiers of tags, which are present in the specific places its sensors cover instead of on tags in the whole space. The second offers application-specific services that are attached to physical entities and places through collections of mobile agents. The third consists of computing devices that

can execute mobile agent-based applications and issue specific events to the agents running in them when location sensors detect the movement of the physical entities or places to which the agents are bound.

4.1 Location Information Server

Each LIS maintains up-to-date information about the identities of the tags within the coverage of its sensors by polling the sensors or receiving specific events issued by them.

Locating Sensor Management

To hide differences among the underlying locating systems, each LIS maps low-level position information from each of the systems into location information in a symbolic location model. An LIS represents an entity's location in terms of the unique identifier of the sensor that detects the tag of the entity. We call each sensor's coverage area a *cell*, as as in the location model of Leonhardt [6]. The current implementation supports four commercial RFID systems: RF Code's Spider system (305-MHz), Alien Technology's RFID system (915-MHz), Philips's I-Code system (13.56-MHz), and Hitachi's mu-chip system (2.45-GHz). The first system provides active RFID tags and its readers can detect RF-beacons issued by tags within a range of 1 to 20 meters. The second provides passive RFID tags and its readers periodically scan for tags present within a range of 3 meters. The third and fourth systems are passive RFID tag systems that can sense the presence of tags within a range of a few centimeters. Although there are many differences between the four RFID systems, our framework hides these differences.

Agent Discovery Mechanism

The framework provides demand-driven mechanisms for discovering the agents and agent host required. Each LIS discovers mobile agents bound to the tags present in its cells and maintains a database by storing information about each agent host and each mobile agent attached to a tagged entity or place. When a LIS detects the presence of a new tag in a cell, it multicasts a query message with the identify of the new tag and its own network address to all the agent hosts in its current sub-network and then waits for reply messages from the hosts. We anticipate one of two possible cases: the tag is attached to an agent host, or the tag is attached to a person, place, or thing but not an agent host. In the first case, the newly arriving agent host sends its network address and device profile to the LIS; the profile describes the capabilities of the host, e.g., input devices and screen size. The LIS stores the profile in its database and forwards a copy of the profile to all agent hosts within the cell. In the second case, agent hosts that have agents tied to the tag send their network addresses and requirements of suitable agents to the LIS; the requirements for each agent specify the required capabilities of the agent hosts that the agent can visit and at which it performs its services. Then, the LIS stores the requirements of the agents in its database and moves the agents to appropriate agent hosts, as discussed below. When the absence of a tag is detected, each LIS multicasts a message with the identifier of the tag and the identifier of the cell to all agent hosts in its current sub-network. Since LISs can be individually connected to other servers, which may be in other sub-networks and with which they exchange information in a peer-to-peer manner, they can discover agent hosts and mobile agents that may be in other sub-networks.

Mobile Agent Navigation

When an LIS recognizes 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 then selects candidate destinations from a set of agent hosts within the cell based on their device capabilities. This framework offers a description language based on CC/PP [12], for specifying the properties of computing devices (vendor, model class of device, e.g., pc, pda, phone, etc., screen size, display colors, CPU, memory, input device, secondary storage, loudspeaker, etc) in XML notation. Each LIS informs each agent of the profiles of the agent hosts that are present in the cell and that satisfy the requirements of the agent, and then the agent can migrate autonomously to the appropriate host.

4.2 Agent Host

Each agent host offers two functionalities: advertisement of its capabilities and a runtime system for executing and migrating mobile agents. When a host receives a query message with the identifier of a newly arriving tag from an LIS, it can respond in one of the following ways: (i) if the identifier in the message is equal to the identifier of the tag to which it is attached, it returns profile information about the agent's capabilities to the LIS. (ii) If one of agents running on its runtime system is tied to the tag, it returns its network address and the requirements of the agent. (iii) If neither case is true, it ignores the message.

The current implementation of this framework is based on a Java-based mobile agent system called MobileSpaces [8]. Each agent host provides a MobileSpaces runtime system built on the Java virtual machine and can move agents to other agent hosts over a TCP/IP connection. The runtime system governs all of the agents inside it and maintains the life-cycle state of each. When the life-cycle state of an agent changes, for example, before or after it is created, terminated, or migrated to another host, the runtime system issues specific events to the agent.

4.3 Mobile Agents

Each mobile agent is executed and migrated in the MobileSpaces runtime system. It is constructed as a collection of Java objects and is equipped with the identifier of the tag to which it is attached. Every agent program must be an instance of a subclass of the abstract class `TaggedAgent` as follows:

```
class TaggedAgent extends MobileAgent
  implements Serializable {
  void go(URL url) throws NoSuchHostException { ... }
  void duplicate() throws IllegalAccessException { ... }
  void destroy() { ... }
  void setTagIdentifier(TagIdentifier tid) { ... }
```

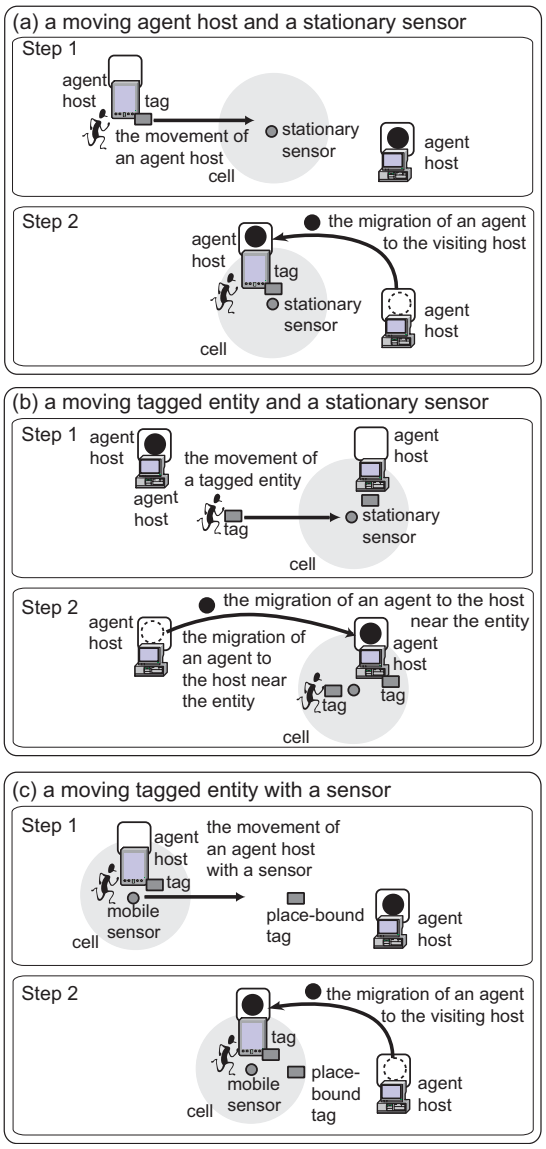

Figure 2: Three types of linkages between physical and logical entities

```
  void setAgentProfile(AgentProfile apf) { ... }
  URL getCurrentHost() { ... }
  boolean isConformableHost(HostProfile hpf) {...}
  ....
}
```

Here are some of the methods defined in the `TaggedAgent` class. An agent executes the `go(URL url)` method to move to the destination host specified as the `url` by its runtime system.[1] 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 called `isConformableHost()`, which the agent uses to decide whether or not the capabilities of the agent hosts specified as an instance of the `HostProfile` class satisfy the requirements of the agent. Each agent can have more than one listener object that implements a specific listener interface to hook certain events issued before or after changes in its life-cycle state or the movements of its tag.

4.4 Current Status

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 host to the destination host recommended by the LIS. This experiment was conducted with two LISs and two agent hosts, each of these was running on one of four computers (1-GHzh Pentium with Windows 2000 and JDK 1.4) that 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 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 the agent from the source host to the destination host, unmarshaling of the agent, and security verification. We believe that this latency is acceptable for a location-aware system used in a room or building.

4.5 Security and Privacy

The framework can be built on many Java-based mobile agent systems with the Java virtual machine. Therefore, it can directly use the security mechanism of the underlying mobile agent system. The Java virtual machine can explicitly restrict agents so that they can only access specified resources to protect hosts from malicious agents. The framework only maintains per-user profile information within those agents that are bound to the user. It promotes the movement of such agents to appropriate hosts near the user in response to the user's movement. Since agents carry the profile information of their users within them, they must protect such private information while they are moving over a network.[2]

5 Location-based Services

This framework can inform mobile agents attached to tags about their appropriate destinations according to the current positions of the tags. It supports three types of linkages between a physical entity such as a person, thing, or place, and one or more mobile agents, as shown in Figure 2.

- The first type of linkage assumes that a moving entity carries more than one tagged agent host and that a space contains a place-bound tag and sensor (Fig. 2 a). When

[1] In MobileSpaces, agents can have higher-level routings among multiple hosts [10].

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

the sensor detects the presence of a tag that is bound to one of the agent hosts, the framework instructs the agents attached to the tagged place to migrate to the visiting agent hosts to offer location-dependent services.

- The second type of linkage assumes that tagged agent hosts and sensors have been allocated (Fig. 2 b). When a tagged moving entity enters the coverage area of one of the sensors, the framework instructs the agents attached to the entity to migrate to the agent hosts within the same coverage area to offer the entity-dependent services of the entity.
- The third type of linkage assumes that an entity carries a sensor and more than one agent host and that a space contains more than one place-bound tag (Fig. 2 c). When the entity moves near a place-bound tag and the sensor detects the presence of the tag within its coverage area, the framework instructs the agents attached to the tagged place to migrate to the visiting agent hosts to offer the location-dependent services of the place.

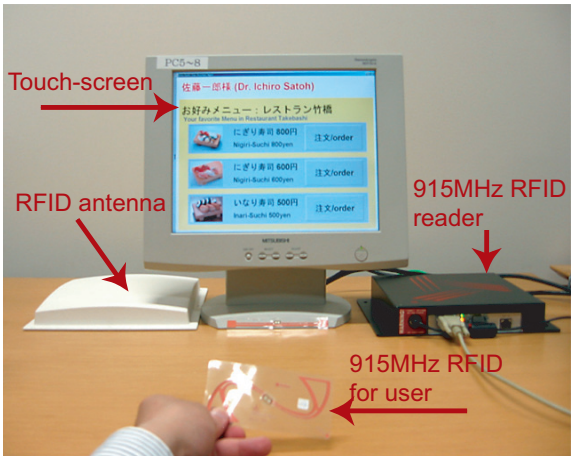

Figure 3: Screenshot of the follow-me user assistant agent that selects user's favorite sushi from menu database of a restaurant in front of the user

6 Applications

This section presents two typical location-based services developed using this framework. The first example corresponds to Fig. 2 (b). This service tracks the current location of a user with a 915-MHz RFID system and provides a user assistant agent that follows users and maintains profile information about them so that the agents can assist them in a personalized manner anywhere. Suppose that a user has a tag and is moving by a restaurant that offers a RFID reader and an agent host with a touch-screen. When the tagged user enters the coverage area of the reader, the framework enables his/her assistant agent to automatically 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 the restaurant menu. It then selects appropriate candidates meals from the menu based on the user's profile information, such as favorite foods and recent during experiences, stored inside the agent. Next, it displays the list of the candiate meals on the screen of the current agent host in a personalized manner. Figure. 3 shows how the user's assistant agent runs on the agent host of the restaurant and seamlessly embeds the pictures, names, and prices of the candidates meals with buttons for ordering them into its graphical user interface. Since a mobile agent is a program entity, we can easily define a more intelligent assistant agent.

Figure 4: screen-shot of map-viewer agent running on tablet PC with positioning sensor

The second example is a user navigation system that assists visitors to a building. Active RFID tags are positioned at several places in the building in the ceilings, floors, and walls. As illustrated in Figure 2 (c), each visitor carries a wireless-LAN- enabled tablet PC equipped with an RFID reader to detect tags. The PC includes an LIS and an agent host. The system initially deploys place-bound agents to hidden computers within the building. When a tagged position is located by the cell of the moving reader, the LIS running on the visitor's tablet PC detects the presence of the tag and detects the place-bound agent tied to the tag. It then instructs the agent to migrate to its agent host and provide the agent's location-dependent services to the host. The system enables more than one agent tied to a place to move to a tablet PC; the agent then returns to its home computer and other agents, which are tied to another place, move to the tablet PC. Figure 4 shows a place-bound agent being used to display a map of its surrounding area on the screen of a tablet PC.

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 [11]. 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] 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.

[3] 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).

[4] 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.

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

[6] 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.

[7] 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.

[8] 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.

[9] I. Satoh: SpatialAgents: Integrating User Mobility and Program Mobility in Ubiquitous Computing Environments, Wireless Communications and Mobile Computing, vol.3, no.4, pp.411-423, Wiley, June 2003.

[10] I. Satoh: Building Reusable Mobile Agents for Network Management, IEEE Transactions on Systems, Man and Cybernetics, vol.33, no. 3, part-C, pp.350-357, August 2003.

[11] I. Satoh: A Testing Framework for Mobile Computing Software, IEEE Transactions on Software Engineering, vol. 29, no. 12, pp.1112-1121, December 2003.

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