A Mobile Intelligent Agent-Based Architecture for E-Business

Zhiyong Weng University of Ottawa, Canada
Thomas Tran University of Ottawa, Canada

ABSTRACT

This article proposes a mobile intelligent agent-based e-business architecture that allows buyers and sellers to perform business at remote locations. An e-business participant can generate a mobile, intelligent agent via some mobile devices (such as a personal digital assistant or mobile phone) and dispatch the agent to the Internet to do business on his/her behalf. This proposed architecture promises a number of benefits: First, it provides great convenience for traders as business can be conducted anytime and anywhere. Second, since the task of finding and negotiating with appropriate traders is handled by a mobile, intelligent agent, the user is freed from this time-consuming task. Third, this architecture addresses the problem of limited and expensive connection time for mobile devices: A trader can disconnect a mobile device from its server after generating and launching a mobile intelligent agent. Later on, the trader can reconnect and call back the agent for results, therefore minimizing the connection time. Finally, by complying with the standardization body FIPA, this flexible architecture increases the interoperability between agent systems and provides high scalability design for swiftly moving across the network.

Keywords: e-business; mobile agents; mobile devices; multi-agent systems; wireless

INTRODUCTION

Many people nowadays use mobile devices such as personal digital assistants (PDA) or mobile phones to access information through the Internet. In addition, they desire to have the ability to participate in e-business anywhere and anytime via their mobile devices. Current e-business applications, such as business-to-consumer (B2C) or Internet-based shopping, are typically developed over the Web for human-computer interaction. These applications require that users must login the intended Web sites from their personal computers or public terminals. Also, users often need to visit lots of sites and are always involved in a time-consuming process. To address these challenges, several wired agent-based e-business systems have been proposed. Kasbah (Chavez & Maes, 1996),

for example, is an electronic marketplace where buying and selling agents can carry out business on behalf of their owners. Nevertheless, these systems do not satisfy the users' mobile demand due to their lack of wireless channels.

This article proposes a feasible architecture that combines agent mobility and intelligence for consumer-oriented e-business applications. It allows a user to create a mobile, intelligent agent via a mobile device, and then launch the agent to the Internet to perform business on the user's behalf. The aspect of mobility enables our architecture to support the agent's migration and the user's mobility (the ability to conduct e-business via mobile devices anyplace and anytime). The mobile agent will migrate from market to market, communicating with different trading agents to find the most appropriate one. Once an appropriate agent is found, it will inform the user of the results. This architecture complements the current Web-based, Internet systems by adding the wireless channel of mobile agents. Our current work focuses on lightweight mobile agents which act on behalf of consumers and participate in consumer-to-consumer (C2C) e-business applications. However, the architecture can be extended to business-to-consumer (B2C) or business-to-business (B2B) applications, as discussed later in the article.

Since personal software agents essentially need to communicate with other agents (to accomplish their designated tasks), they have to comply with a set of standards concerning the agent communication language and the protocols to be used. Although there is currently no universally accepted set of standards for developing multi-agent systems, the Foundation for Intelligent Physical Agents (FIPA), which aims at providing one language commonly understood by most agent-based systems (FIPA, 2006), is obtaining a growing acceptance. With FIPA becoming a de facto standard in this field, the architectures such as JADE (Java Agent Development Environment) have become available to allow for the implementation of a FIPA-compliant multi-agent system such as our proposed architecture (Chiang & Liao, 2004).

It should be noted that mobile devices suffer not only from limited battery time, memory, and computing power, but also from small screen, cumbersome input, and limited network bandwidth and network connection (Wang, Sørensen, & Indal, 2003). The proposed architecture, by making use of mobile agent technology, offers a solution to those problems. That is, after creating and initializing a mobile agent to act on the user's behalf, a user can disconnect the mobile device from the server. The user only needs to reconnect later on to recall the agent for results, hence minimizing the use of resources. In addition, mobile agent technology also addresses such challenges as increased need for personalization, high latency, demand for large transfers, and disconnected operation (Kotz & Gray, 1999).

The remainder of this article is organized as follows: the second section introduces background knowledge and related work. The third section illustrates the proposed architecture. The fourth section shows an implementation of the proposed architecture. The fifth section discusses some existing problems and future works. The sixth section concludes the article.

BACKGROUND AND RELATED WORK

Mobile Agent Paradigm

An intelligent agent is a piece of software, which differs from the traditional one by having such features as being autonomous, proactive, social, and so on. One of these characteristics is mobility, that is, the agents' ability to migrate from host to host in a network. Mobile agents are defined as programs that travel autonomously through a computer network in order to fulfill a task specified by its owner, for example, gathering information or getting closer to the required resources to exploit them locally rather than remotely. A mobile agent is not bound to the system on which it begins execution, and hence can be delegated to various destinations. Created in one execution environment, it has the capability of transporting its state and code with it to another host and execute in the same execution environment in which it was originally created. Several mobile agent systems have been designed in recent years. Telescript (White, 1996) is the first commercial mobile agent system developed by General Magic. Telescript provides transparent agent migration and resource usage control. Aglets from IBM (Lang & Oshima, 1998) is also a mobile agent system based on the concept of creating special Java applets (named aglets that are capable of moving across the network). JADE (Bellifemine, Caire, Trucco, & Rimassa, 2006) is one of the agent development tools that can support efficient deployment of both agents' mobility and intelligence in e-business applications. As a middleware implemented in Java and compliant with the FIPA specifications, JADE can work and interoperate both in wired and wireless environments based on the agent paradigm. JADE supports weak mobility; that is, only program code can migrate while no state is carried with programs. NOMADS (Suri et al., 2000) supports strong mobility and secure execution; that is, the ability to preserve the full execution state of mobile agents and the ability to protect the host from attacks.

Recently, mobile agents have found enormous applications including electronic commerce, personal assistance, network management, real-time control, and parallel processing (Lange & Oshima, 1999). Kowalczyk et al. (2002) discuss the use of mobile agents for advanced e-commerce applications after surveying the existing research. There are many advantages of using the mobile agent paradigm rather than conventional paradigms such as client-server technology: reduces network usage, introduces concurrency, and assists operating in heterogeneous systems (Lange & Oshima, 1999).

Related Work

Mobile agents have been recognized as a promising technology for mobile e-business applications. The interest of developing mobile agent systems for mobile devices has increased in recent years. Telescript describes a scenario in which a personal agent is dispatched to search a number of electronic catalogs for specific products and returns best prices to a PDA from where it starts (Gray, 1997). An integrated mobile agent system called Nomad allows mobile agents to travel to the eAuctionHouse site (http://ecommerce.cs.wustl.edu) for automated bidding and auction monitoring on the user's behalf even when the user is disconnected from the network (Sandholm & Huai, 2000). They aim at reducing network traffic and latency.

Impulse (2006) explores a scenario in which e-business meets concrete business through a system of buying and selling agents representing individual buyers and sellers that carry out multiparameter negotiation and running on the wireless mobile devices. Impulse deploys personal agents on mobile devices to help users seek agreement on purchase terms. However, these personal agents are directed to move online to participate in negotiations, and hence resulting in potentially long-time connection with the Internet. We also think that the Impulse system was designed with a single communication protocol for all agents. This presents drawbacks due to the heterogeneity of exchanged information and leads to an inflexible environment, which can only accept those agents especially designed for it. Agora (Fonseca, Griss, & Letsinger, 2001) is a project conducted at HP Labs to develop a test-bed for the application of agent technology to a mobile shopping mall. A typical scenario consists of mobile shoppers with PDAs interacting with store services while in the mall, on the way to the store, or in the store itself. The Zeus agent toolkit, developed by British Telecommunications, was used to implement all agents in the Agora project. Only the infrastructure agents speak the FIPA Agent Communication Language (ACL), causing the architecture to conform partly to FIPA, although more effort in conformance is needed. The purpose of the Agora project is to gain experience in agents' communication protocols and to realize the significance of architectural standards.

It has been shown that modern agent environments such as JADE could be easily scaled to 1,500 agents and 300,000 messages (Chmiel et al., 2004). Thus, it is now possible to build and experiment with large-scaled agent systems. Moreno et al. (2005) use JADE-LEAP (JADE Lightweight Extensible Agent Platform) to implement a personal agent on a PDA. This agent belongs to a multi-agent system that allows the user to request a taxi in a city. The personal agent communicates wirelessly with the rest of the agents in the multi-agent system, in order to find the most appropriate taxi to serve the user. IMSAF (Chiang & Liao, 2004) is an architecture designed to fulfill the requirements of Impulse-introduced mobile shopping and implemented using JADE-LEAP tools. LEAP can also be used to deploy multi-agent systems spread across mobile devices and servers; however, it requires a permanent bidirectional connection between mobile devices and servers. Considering the current expensive connection fees for cell phones, such a required permanent connection is not affordable for consumers in practice.

In contrast to the above works, we are motivated to propose a mediator-based architecture that attempts to enable users' wireless participation in several e-marketplaces through their mobile devices. The mobile agents can move across the network and perform trading tasks on behalf of their users when the users are disconnected from the network. We believe that it is important to consider the limitations of mobile devices, such as low-battery, low bandwidth, high latency, limited computing ability, and expensive connection fees. The fact that consumers in our physical world may need to access the worldwide markets and distributed e-business environments requires the agents to operate in heterogeneous and dynamic environments as well as to talk a common language. By complying with the FIPA specifications, the proposed architecture provides an interoperable solution to allow users dynamically to connect to the network by means of their mobile

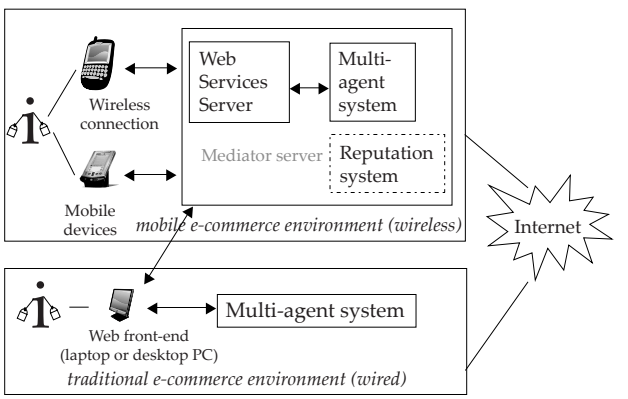
devices only when needed. Also, the mobile devices will not suffer those limitations mentioned above. In addition, the benefit of using mobile agents to a user becomes more obvious in our architecture if the user has a mobile phone and is interested in minimizing expensive connection costs. The next section explains the architecture in more detail.

SYSTEM ARCHITECTURE

Overview

Figure 1 shows a distributed C2C wireless e-business environment and a traditional wired e-business one. A consumer can connect a mobile device, such as a PDA or mobile phone, to the mediator server through wireless connection and then send a request for creating a mobile buying or selling agent to undertake a specific business task (e.g., auction bidding) on the user's behalf. A personal agent that resides on the mobile device is needed to interact directly with the consumer and to consider the consumer's personal preferences. Considered as a true representative of the consumer, the personal agent represents the consumer's interests and allows the consumer to have a choice of dispatching either a buying or a selling agent. The mediator server sits in the fixed network and provides services such as generating mobile agents according to consumers' requests. After being created, the mobile agents will autonomously travel to multiple agent-based servers on the Internet. The agent-based servers offer places for selling and buying agents to meet and negotiate with one another. The proposed mediator server contains two main components: the Web services server, which facilitates mobile agents to interface with other agents, and the multi-agent system, which manages the agents and plays the role of a marketplace similar to an agent-based server. An additional component, a reputation system, will be necessary in our architecture. Using this reputation system, agents could sign binding contracts and check user's credit histories and reputations. The trust problem will be further studied in future research (e.g., Jøsang & Ismail, 2002 present a Beta Reputation System). Also, the mediator server provides a Web-based interface, and as shown in Figure 1, a consumer can also connect a laptop or a desktop PC to the network and launch an

Figure 1. Distributed e-business environment



agent to execute on the mediator server.[1] In this article, we focus on the electronic trading of second-hand products for owners of mobile devices.

The main idea is that a consumer will request the mediator server to create a buying or selling agent and then dispatch it to agent-based servers on the Internet. The main operation that occurs in an agent-based server is price negotiation where buying agents negotiate price with selling agents. According to the consumer's preferences, the buying agent may travel to different e-market sites known by the white-page agent[2] to seek goods, when the consumer desires to conduct a global multiple markets comparison. The W3C's XML schema specification (www.w3.org/XML/Schema) provides a standard language for defining the document structure and the XML structures' data types. The consumer's preferences can be represented in an XML format. In a real business situation, we would have to ensure that messages are reliably delivered to the mediator server from the personal agents. Although this communication protocol's reliability is not detailed in our architecture currently, we could use a reliable transport at the very least, such as Reliable HTTP (HTTPR) (Todd, Parr, & Conner, 2005), for the communication between the personal agents and the mediator server. Another consideration is to encrypt the communication. Encryption technologies can also help ensure that even intercepted transmissions cannot be read easily.

A Scenario of Our Architecture

To understand the environment best, let us consider a typical scenario taken from daily life, where two hypothetical customers, named Mary and Tom, try to participate in an eBay-like auction. Mary wants to sell her used Sony MP3 player. At her office, she initiates a selling agent from a PDA, through a wireless LAN connection with the mediator server in the building. Then this selling agent lives in the server and waits for potential shoppers. Due to some unpredictable event, Mary may have to leave her office and cannot access the selling agent via her PDA (as there may be no available wireless LAN network coverage). However, she will be able to reconnect later on.

Haphazardly, Tom enters his buying preferences into his Java-enabled mobile phone, trying to buy a second-hand Sony MP3 player under a maximum price. The personal agent on his mobile phone establishes a connection with the mediator server and asks the server to launch a mobile buying agent according to his preferences. Then Tom disconnects his cell phone from the server. The mobile agent knows where and how to migrate, as instructed in the migration itinerary. As days pass, while this buying agent is roaming around the Internet, it enters into Mary's mediator server and searches for services provided. After the negotiation between the selling agent and buying agent, they reach an agreement on the item and price. With that, the buying agent will return to its host server and send a SMS (Short Message Service)-based notification to the personal agent running on Tom's mobile phone, about the potential seller gathered from the Internet. Also the selling agent sends an e-mail-based notification to the personal agent running on Mary's PDA. Finally, things left to Mary and Tom seem to be simple and easy since they could have either the cell phone number or the e-mail address from the information reported by their personal agents, respectively. As we have seen, mobile consumers only need a small bandwidth connection twice, once

for initiating a migrating mobile agent and once for collecting the results when the task is finished.

Architecture Description

We explain how the whole system works in this section. Figure 2 illustrates the system architecture and the operation process. As shown in Figure 2, mobile devices are supported by personal agents and connected to the mediator server via a wireless connection. A personal agent is a static agent running on a mobile device and offers a graphical user interface (GUI) for its user to communicate with the system. The mediator server is connected to the Internet where other mediator servers or other FIPA-compliant systems exist. In the mediator server, a servlet answers any requests from the personal agent and is linked to the behavior of a proxy agent[3] in charge of handling the requests. The proxy-agent interfaces with the servlet and constructs a bridge between the Web service server and the multi-agent system. Each instance of the behavior[4] connects not only to the AMS agent (Agent Management Service as defined in FIPA, i.e., the white-page agent mentioned above), asking for the creation of a buying or selling mobile agent in the multi-agent system as well as providing a response, but also connects to the agent DF (Directory Facilitator as defined in FIPA, i.e., the yellow-page agent), retrieving the list of agents advertising services with the DF. In this architecture, the multithreaded-servlet server is mirrored by a multibehavior proxy agent to allow for handling multiple requests in parallel.

As illustrated in Figure 2, the procedures from (1) to (6) depict how a buying or selling mobile agent is created by a user according to preferences:

1. At the first step, the user configures the preferences via the personal agent (residing in the mobile device). The personal agent then sends an XML-based request to the mediator server.
2. An instance of the servlet accepts the request and communicates with the proxy agent.
3. The proxy agent cooperates with the AMS agent who lives in the main container of the JADE platform to create a buying or selling mobile agent.
4. If the buying or selling agent is created successfully in the container, it might be mobilized to other systems to undertake the user's task.
5. and (6) The personal agent receives a response from the proxy agent via the servlet and informs the user of the relevant mobile agent being created.

The above is an asynchronous process after which the user can disconnect from the network at will. Even if the user decides to disconnect from the network, the user will still receive an SMS-based notification from the mediator server via an interface with the wireless carrier, or an e-mail-based notification from the mediator server via an interface with a mail server, as long as the user reconnects to the network.

The mediator server provides the required support for the creation of mobile agents, messaging among agents, agent migration facility, collaboration, protection, destruction, and control of mobile agents. Mobile agent platforms such as JADE have been proposed to provide the supporting environment. Obviously, any multi-agent system can be used here as long as it provides the required support.

Different Types of Agents in Our Architecture

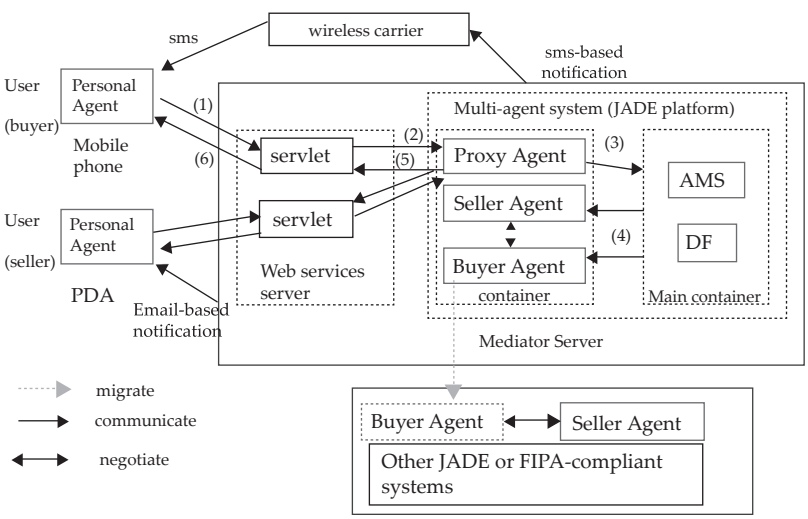
The following agents co-exist in our architecture: personal agents, proxy agents, buying or selling agents, yellow-page agents, and white-page agents. Among them, only buying or selling agents are mobile agents, while personal agents and proxy agents are stationary agents. Both the yellow-page and white-page agents are fixed on a component of the mediator server. Details of these agents are described as follows:

A personal agent is a stationary agent that runs on a user's mobile device and provides a graphical interface to allow the user to configure a mobile buying or selling agent (from the mobile device). When starting the personal agent on the mobile device, the user can choose either to initiate a new mobile agent or to recall a previous mobile agent. One may argue that such a personal agent is nothing more than an interface. From the agent's viewpoint, however, the personal agents are able to autonomously communicate with the proxy agent which is running in the mediator server.

A proxy agent is also a stationary agent which links the multi-agent system to the Web service server. It is one of the agents that is always up and running in the multi-agent system. The proxy agent cooperates with the AMS (white-page) agent to create a mobile buying or selling agent for each user. There is only one proxy agent per mediator server due to its unique multibehavior ability.

A yellow-page agent (such as the DF agent in the JADE platform) provides the service of yellow pages, by means of which an agent can receive information about available products or find other agents providing necessary services to achieve its goal. A white-page agent (like the AMS agent in the JADE platform) represents the authority and provides naming services. It stores information about addresses and identifiers of all agents in the system. In our architecture, sellers have permission to advertise their products; and buyers are allowed to query the sellers which post the products they are looking for. Selling agents update yellow pages by publishing

Figure 2. System architecture and process



their services via the yellow-page agent. Buying agents query relevant services from the yellow-page agent. Both buying and selling agents update white pages by registering in or deregistering from the system. They communicate with each other via querying agent's information from the white-page agent.

Both buying and selling agents are mobile agents, which are also called *service agents*. A service agent is the counter part of a personal agent and is involved in the migration from host to host on the Internet. A service agent first negotiates with other service agents in the same host mediator server before migrating among multiple Web sites to talk to other service agents, provided that they can talk a common language.

To demonstrate a useful mobile agent system, we present a prototype for buying and selling agents, with attributes depicted in Table 1. This means that a user will configure a mobile buying or selling agent on a mobile device, precisely according to the characteristics in Table 1.

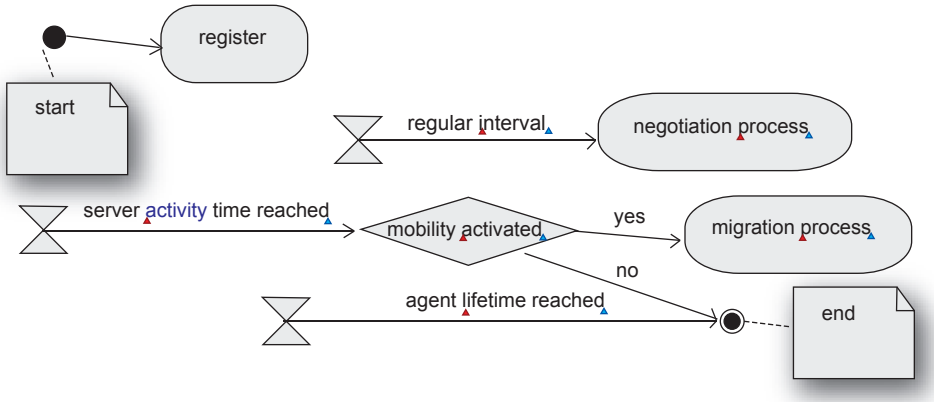
Behaviors of Mobile Agents

As illustrated in Figure 3, a mobile (buying or selling) agent starts with its registration in the system and ends with a timeout of its lifetime. There are three time events that indicate the behaviors of a mobile agent: (1) the agent starts its negotiation process at a regular interval (e.g., every minute); (2) the agent starts its migration when activity time per server is reached; and (3) the agent ends its life cycle when its lifetime is exhausted. An argument may arise; how can one be sure that the mobile agent will be terminated according to the parameter and lifetime, as users prefer? This parameter may be changed by a third party (including the mediator server). The assumption we made is that the mobile agent can be protected from the attacks (e.g., from the host or other agents) once a future security mechanism is imposed on our architecture.

Table 1. Attributes of a mobile agent

Attribute	Description
Agent type	The agent type that a user can select, that is, either a buying agent or a selling agent
Agent server	The configuration of the mediator server address.
User id	The user identification which can be email address, cell phone number, or IMEI (International Mobile Equipment Identity).
Quantity	Quantity of the predefined product.
Price	For a buying agent, this is the maximum price that the agent can bid: for a selling agent, this is the minimum price that the agent can accept.
Current Price Inquired	For a buying agent, this is the best price offer colledted from the Internet.
Lifetime	The total time an agent can be away before being recalled or terminated.
Mobility	Specification of whether a user desires to enalbe the agent's migration ability (i.e., in the context of a local, single or global, multiple market comparison).
Server Activity Time	The time an agent can spend on each server before migrating ot another.

Figure 3. Activity diagram of mobile agent



(The security problem is discussed in the Discussion and Future Work section.)

Negotiation Process

The proposed interaction between agents complies with the FIPA-Contract-Net Protocol (FIPA, 2006). This protocol allows a buying agent (initiator) to send a call for proposals (CFP) to a set of selling agents (responders), evaluate their proposals, and then accept the most preferred one (or even refuse all of them). Both initiators and responders should register in the system before they negotiate with each other.

In this article, we consider a classical situation in which a selling agent offers a single item to the highest bidder (similar to eBay), and the simplest type of bid is an offer to buy or sell one unit at a specified price. As shown in Figure 4, the buying agent sends a CFP to all the available selling agents (obtained from the yellow-pages service). After receiving the message, a selling agent can send the buying agent a proposal with the price for the product. If the product is not available or sold, it does not need to send any proposal. The buying agent will place a purchase order if the offer price is within the maximum price that the customer has specified. Results of price negotiations are sent back to the personal agent and showed in a graphical interface to the user. Since the system is fully asynchronous, an intention to make a purchase does not have to lead to a successful transaction. By the time the offer is made, other buying agents may have already purchased the last available item.

Agent Migration

The general process of migration is depicted in Figure 5. An agent starts its migration from its host server (i.e., the mediator server) with the itinerary list acquired from the host. We assume that there are n servers, which will be visited by the agent in sequence. In each server, two time events happen resulting in two actions respectively: if the agent reaches its lifetime, it will return to its host where it was created, and then end the migration process; if the agent exhausts its server activity time, it will migrate to the next server. Additionally, before the agent migrates to the next server, it should

Figure 4. Negotiation process

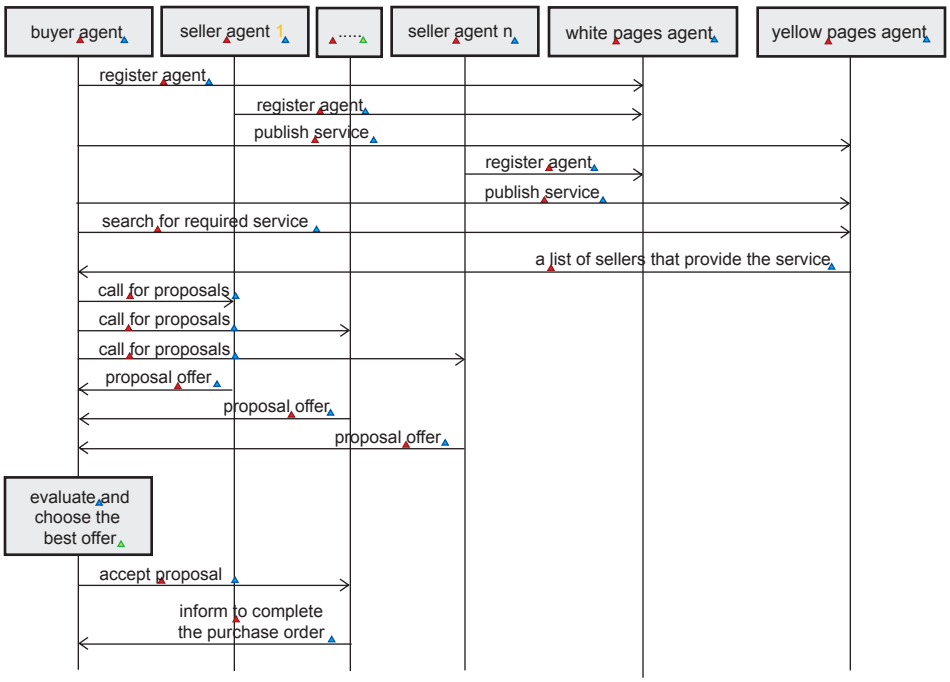
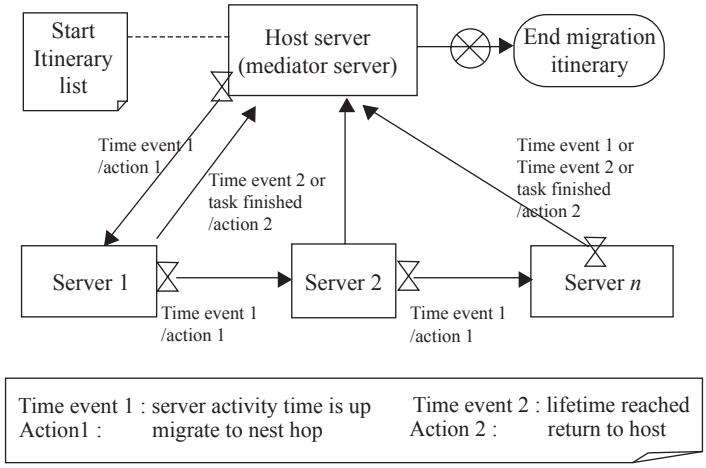


Figure 5. Agent migration process



also make the decision if it has fulfilled the task at the current server. As we know, a task is finished when an agent receives an acceptable offer from another agent. The migration process actually describes a scenario of price comparison (finding a price less than a buyer's reservation price for buying, or searching for a price greater than a seller's reservation price for selling). The agent may access its host server repeatedly during its lifetime and updates its itinerary list every time when visiting its host server. One interesting problem here is how the mediator server maintains the itinerary list that includes a series of service-providing servers to be visited by the agent. Curbera, Duftler, Khalaf, Nagy, Mukhi, and Weerawarana (2002) state that "several individual companies and industry groups are starting to use 'private' UDDI directories to integrate and streamline access to their internal services" (p. 90). UDDI (Universal Description, Discovery and Integration) (UDDI, 2006) enables businesses to publish service listings and to discover each other. We assume that the white-page agent can interact with the UDDI server to obtain other service-providing servers' addresses (the feasibility of this function will be further studied) and therefore mobile agents can update the itinerary list during their migration. Only the mobile agents, which are originally created in this mediator server, are allowed to access this resource (a list of servers).

System Implementation

We have implemented a simple prototype to evaluate the concepts proposed in our architecture, using the Java programming language. Figure 6 shows the screenshots of a personal agent and a JADE-based multiagent system, respectively. The personal agent was developed as a J2ME MIDlet[5] application that offered a graphical interface for its user to initiate or recall the mobile agent, and to dialogue with the mediator server. The mediator server played an important role in our architecture, running a Tomcat Apache Servlet Engine on a JADE platform. JADE is an open-source with good scalability, one of the best modern agent environments compliant with FIPA. As shown in Figure 6, there are two containers on the JADE system, Main-container and Container-1. Main-container holds the basic management agents defined by FIPA (AMS, DF, and RMA, which manages the GUI of the JADE platform). The proxy agent, buying agents, and selling agents run in Container-1. We can deploy the mediator-based architecture in one or several PCs.

The Web services architecture communications are based on JSR172, J2ME Web services, which include two independent parts: the JAX-RPC and JAXP. XML is chosen as the standard way for clients to interact with backend servers so as to use the remote services. J2ME JAX-RPC APIs subset solves how to access the SOAP/XML Web services and JAXP APIs subset solves how to process the XML messages. Messages exchanged by agents in the multiagent system have a format specified by the ACL language defined by FIPA for agent interoperability.

As shown in Figure 7, we deployed three servers in the Local Area Network, installed J2ME MIDlet in two mobile phone simulators, provided one GUI for the Web-based seller, and simulated a simple used-item electronic trading scenario similar to the one we described in A Scenario of Our Architecture section previously. The mobile phone emulator is a tool provided by the Sun J2ME wireless toolkit 2.2. Both mediator servers deployed the Tomcat server and

Figure 6. Screenshots of a personal agent and the JADE platform

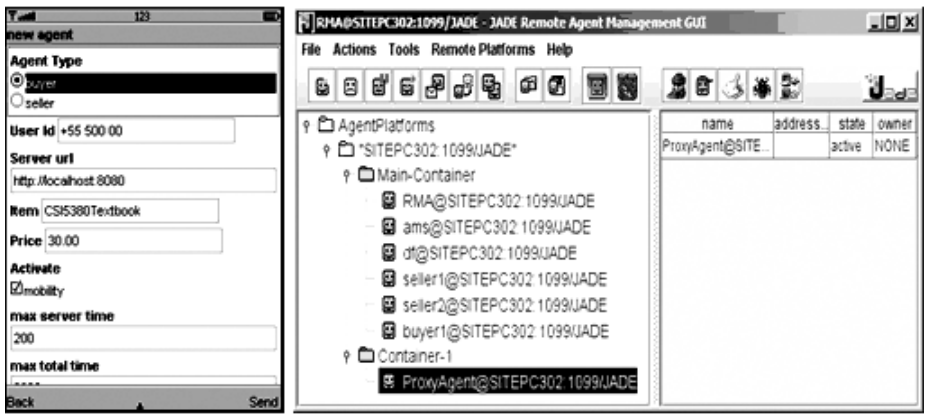
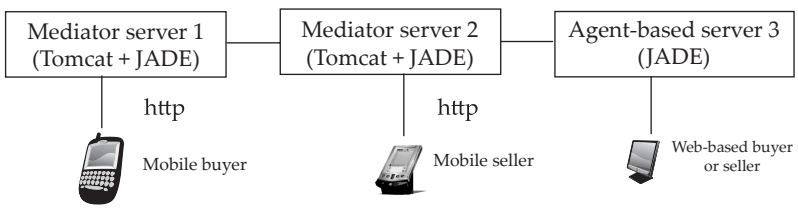


Figure 7. Experiment environment



the main container of JADE platform was initialized. The third computer played the role of an agent-based marketplace on the Internet. For the buyer's emulator, the user activated the mobility of the buying agent, but it was not the case for the seller's emulator. We observed the following results:

- Mobile users can connect to mediator servers via HTTP and initiate mobile buying or selling agents in the mediator server.
- Mobile users do not need to instruct their mobile agents of what to do after configuring their preferences.
- Mobile users can add new items anytime and relevant mobile agents will be created to handle the trading of these new items respectively.
- Mobile users can kill their mobile agents to cancel their tasks by sending instruction to their personal agents.
- Mobile agents are active in their servers within the specified server activity time and then migrate to other servers.
- Buying agents can reach agreements with selling agents when the required item and price are matched. Mobile users then receive text messages from their agents, displayed on the screen of the simulators.
- Mobile agents end their life cycles when finishing their tasks.

As confirmed by the experiments, mobile users connect to their servers only when they need to add new items or to cancel their tasks. This obviously results in such

benefits as reduced bandwidth utilization, increased battery life for mobile devices, and no complicated computation conducted in mobile devices. Also, mobile agents can move to various servers to negotiate autonomously, and mediator servers can accept mobile agents from outside their systems. This feature enables users to participate in multiple markets on the Internet.

In particular, we observed the migration process of a mobile agent. Mobile agents should be active in their servers within a specified time and migrate among the servers. Thus, we developed a scenario where we supposed that a buying agent started from Server1 and continued searching for the required product or service in Server2 and Server3. We set up two parameters for this mobile agent: Maximum server active time was set to 100 seconds and total lifetime was set to 850 seconds. As expected, the buying agent contacted the other agents in Server1 and then migrated to Server2 after approximately 100 seconds. Similarly, the buying agent communicated with other agents in Server2 and then traveled to Server3. The same things happened in Server3. Because there were no more sites to be visited, the buying agent migrated back to Server1, ending its first round of migration. The second round was started since the total lifetime was not reached. We assumed that no sellers offered the required product or service to this buying agent. With the time elapsed, the buying agent was in its third round and roamed into Server2. At this stage, the buying agent used up its lifetime of 850 seconds and predicted an ending of its life cycle. Therefore it migrated back to the host Server1, even though the third round trip was not finished. In another scenario, we used the same parameters for the buying agent, except that the total lifetime was enlarged to 1,000 seconds. The difference was that we dispatched a selling agent in Server2 at the moment the buying agent was ready to launch its third round trip. This selling agent offered exactly the service that the buying agent needed. As we expected, the two agents met and reached an agreement after negotiating with each other. This experiment confirmed that after completing its task, the buying agent migrated back to Server1, regardless of its remaining lifetime that had not yet been exhausted.

DISCUSSION AND FUTURE WORK

In this article, we propose a feasible mobile agent architecture that assists users in C2C e-business. It enriches the resources for users to perform comparison shopping activities at the point of purchase. Users' mobile devices connect to the network only when needed, thus making efficient use of limited bandwidth and reducing the network traffic. In addition, it helps cell phone users save money from their expensive bills. At any time, users may add items via their personal agents and specify their preferences such as time limit and preferred price for trading. Through the negotiation process between mobile buying and selling agents, users also gain valuable information for making trading decisions.

Our proposed architecture is extensible: On the one hand, XML-based communication is used to enhance extensibility; on the other hand, the architecture could be easily extended to B2C, or even B2B business models. That is, not only individuals but also business companies can be attached to the architecture. With mobile phones and PDAs already being used as extended enterprise tools, business companies, such as retailers and suppliers, can publish their

products and/or services on their servers via mobile devices. As long as these businesses take part in our architecture parties, they could benefit from the automatic discovery of other business partners. Also, it is possible for businesses, especially for retailers, to sell their products to potential buyers in the manner described in the proposed architecture as an extra way to their traditional ones. In this sense, our architecture is an integration model of C2C, B2C, and B2B e-business. Nonetheless, using mobile devices for complex tasks can be quite frustrating (e.g., difficult to enter data), so probably people will not use it. An idea is to incorporate targeted messaging or advertising into our model, where businesses could send a message to users who are physically located in their vicinity. Agents could negotiate a transaction, and the buyer would already be located nearby to complete the purchase and pick up the item.

Currently, we present a conceptual framework that needs to be refined. Using this work as a starting point, we have outlined a number of future research directions:

(1) Negotiation protocols do not have to be hard-coded into the agents. Instead, mobile agents can adapt to any intelligent negotiation strategies when they arrive at a new remote location. Thus, our architecture paves the way for future research in which more general architectures can be explored to allow mobile agents to participate in a variety of negotiation protocols, such as factor negotiation (price, quality, delivery time, etc.), electronic contracting, and so on. Currently, the negotiation strategy module consists of only a purchase determined by price (agents seek a preferable price by a fixed amount). FIPA defines auction protocols (e.g., Dutch and English auctions) as well as simpler strategies such as fixed pricing, fixed pricing with a discount, and so on. We will add them into the negotiation protocols in our future research.

(2) Items are described only by their names. Obviously, other attributes, such as color, age, terms of warranty and delivery should also be considered. We believe that ontologies can help to solve this problem. It should be noted that the small screen of mobile devices will bring inconvenience to users when they specify many attributes of an item. A possible solution is to make use of the persistent memory of mobile devices to store the users' preferences.

(3) Mobile agent technology currently has some limitations, such as identity management, fault tolerance, protection of agents, and resource security. These limitations have brought up some concerns about the practical utilization of mobile agents. For example, in the area of security, e-business applications are often involved with money and thus users may hesitate to use mobile agents, unless mobile agents are secure enough to be trusted.

In the situation presented in this article, the mobile agents representing different buyers or sellers migrate over the Internet and then execute themselves on remote computers. These mobile agents are thus exposed to open environments and may become vulnerable. Since the mobile agents execute on unknown computers and interact with unknown agents, a reliable security infrastructure is vitally needed for the design of the system. The mobile agents must be able to deal with situations where they have been shipped off to the wrong address or to a hostile environment (Neuenhofen & Thompson, 1998). Listed below are some possible security concerns:

- Malicious mobile agents can try to access services and resources without

adequate permissions. In addition, a malicious agent may assume the identity of another agent in order to gain access to platform resources and services, or to cause mischief or even serious damage to the platform.

- Mobile agents may suffer eavesdropping attack from other mobile agents. A malicious agent can sniff the conversations between other agents or monitor the behavior of a mobile agent in order to extract sensitive information from it.
- Mobile agents may suffer alteration attack from malicious hosts. To execute the agent and update its state, the host must definitely be capable of reading and writing the agent. A malicious host may steal private information from the agent or modify the agent to compute the wrong result or to misbehave when it jumps to another site.

Current research efforts in the area of mobile agent security adopt two different perspectives (Kotz, 2002): First, from the platform perspective, we need to protect the host from malicious mobile agents (such as viruses and Trojan horses) that are visiting it and consuming its resources. Second, from the mobile agent perspective, we need to protect the agent from malicious hosts. There are many mechanisms to protect a host against malicious agents. Digital signatures and trust management approaches may help identify the agent and evaluate how much it should be trusted. The malicious host problem, in which a malicious host attacks a visiting mobile agent, is the most difficult problem. We found in the literature some works on powerful techniques such as Sandboxing and Proof-Carrying Code (PCC). Sandboxing (Wahbe, Lucco, Anderson, & Graham, 1993) is a software technique used to protect a mobile agent platform from malicious mobile agents. PCC (Lee & Necula, 1997) introduces the technique in which the code producer is required to provide a formal proof that the code complies with the security policy of the code consumer. Therefore, we envisage that the security of mobile agents is an important issue that will encourage techniques and mechanisms for e-business in the future.

CONCLUSION

We propose in this article an e-business architecture that allows traders to do business at remote locations by means of mobile intelligent agents. Our architecture, which adheres to standardization efforts in the multi-agent field such as FIPA paves a possible way towards a near future when mobile buying (and selling) agents can smoothly travel among different agent-based marketplaces to carry out tasks on their users' behalves. Our purpose of presenting this idea is to improve our understanding of the value of mobility and to encourage the conceptual construction of a global community. We do not claim that buyers and sellers around the world would have to buy into this to make it work, and that worldwide C2B e-commerce would be revolutionized thereby. In practice, however, we hope that our work would be useful on a smaller scale and lead to new investigations that may result in new solutions to the problems we addressed. Our proposed architecture, aimed at providing new capabilities for advanced e-business solutions, employs an approach that integrates intelligent and mobile agents. Intelligent agents can provide automation support for decision-making tasks, while mobile agents can extend that support by allowing users to participate in several

marketplaces in a networked e-business. We believe that intelligent and mobile agent technology is also a promising solution to the problems of low speed, high latency, and limited computing ability that the current wireless network is facing.

REFERENCES

Bellifemine, F., Caire, G., Trucco, T., & Rimassa, G. (2006). JADE programmer's guide. Retrieved July 7, 2007, from http://jade.cselt.it/docs

Chavez, A., & Maes, P. (1996). Kasbah: An agent marketplace for buying and selling goods. In *Proceedings of the 1st International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology*, London, United Kingdom.

Chiang, H., & Liao, Y. (2004). An agent-based architecture for impulse-induced mobile shopping, *Computer and Information Technology*.

Chmiel, K., et al. (2004). Testing the efficiency of JADE agent platform. In *Proceedings of the 3rd International Symposium on Parallel and Distributed Computing* (pp. 49-57). IEEE Computer Society Press.

Curbera, F., Duftler, M., Khalaf, R., Nagy, W., Mukhi, N., & Weerawarana, S. (2002, March-April). Unraveling the Web services web: An introduction to SOAP, WSDL, and UDDI. *IEEE Internet Computing, 6*(2), 86-93

FIPA. (2006). Retrieved July 7, 2007, from http://www.fipa.org

Fonseca, S., Griss, M., & Letsinger, R. (2001). *An agent-mediator e-business environment for the mobile shopper* (HP Tech. Rep. No. HPL-20010157).

Gray, R.S. (1997). Agent Tcl. *Dr. Dobb's Journal*, pp. 18-26.

Impulse. (2006). Retrieved July 7, 2007, from http://agents.media.mit.edu/projects/impulse/

Jøsang, A., & Ismail, R. (2002, June). The Beta Reputation System. In *Proceedings of the 15th Bled Electronic Commerce Conference*, Bled, Slovenia.

Kotz, D. (2002). Future directions for mobile agent research. *IEEE Computer Science*.

Kotz, D., & Gray, R. (1999). Mobile code: The future of the Internet. In *Proceedings of Autonomous Agents'99: Workshop on Mobile Agents in the Context of Competition and Cooperation*.

Kowalczyk, R., et al. (2002). Integrating mobile and intelligent agents in advanced e-business: A survey. In *Proceedings of Agent Technologies, Infrastructures, Tools, and Applications for E-Services, NODe'2002 Agent-Related Workshops*, Erfurt, Germany.

Lange, B.D., & Oshima, M. (1998). *Programming and deploying Java mobile agents with aglets*. Addison-Wesley.

Lange, D.B., & Oshima, M. (1999). Seven good reasons for mobile agents. *Communications of the ACM*.

Lee, P., & Necula, G. (1997). Research on proof-carrying code on mobile-code security. In *Proceedings of the Workshop on Foundations of Mobile Code Security*.

Moreno, et al. (2005). Using JADE-LEAP to implement agents in mobile devices. Retrieved July 7, 2007, from http://www.zdnet.de/itmanager/whitepapers

Neuenhofen, K.A., & Thompson, M. (1998). Contemplations on a secure marketplace for mobile Java agents. In K.P. Sycara & M. Wooldridge (Eds.), *Proceedings of Autonomous Agents 98*, Minneapolis, Minnesota. New York: ACM Press.

Sandholm, T., & Huai, Q. (2000). Nomad: Mobile agent system for an Internet-based auction house. *IEEE Internet Computing*, pp. 80-86.

Sun. (2006). Java. Retrieved July 7, 2007, from http://java.sun.com/javame/

Suri, N., et al. (2000). NOMADS: Toward a strong and safe mobile system. In *Proceedings of the 4th International Conference on Autonomous Agents* (pp. 163-164). New York: ACM Press.

Todd, S., Parr, F., & Conner, M. (2005). An overview of the reliable HTTP protocol. Retrieved July 7, 2007, from http://www-128.ibm.com/developerworks/webservices/library/ws-phtt/

UDDI. (2006). Retrieved July 7, 2007, from http://www.uddi.org/

Wahbe, R., Lucco, S., Anderson, T.E., & Graham, S.L. (1993). Efficient software-based fault isolation. In *Proceedings of the 14th ACM Symposium on Operating Systems Principles* (pp. 203-216).

Wang, A.I., Sørensen, C.F., & Indal, E. (2003). A mobile agent architecture for heterogeneous devices. *Wireless and Optical Communications*.

White, J.E. (1999). Telescript technology: Mobile agents. In Mobility: Processes, computers, and agents (pp. 460-493). New York: ACM Press/Addison-Wesley.

ENDNOTES

1 In the experiment, we developed a GUI in the mediator server for users to launch a buying or selling agent.

2 The white-page agent maintains different service provider sites. Section 3.4 will describe this agent in more detail.

3 Detailed description of the proxy agent is provided in Section 3.4.

4 In an object-oriented context, a behavior is an inner class of the proxy agent.

5 MIDlet is a Java program generally running on a cell phone, for embedded devices, more specifically the Java ME virtual machine.

Zhiyong Weng is a graduate student in the school of information technology and engineering at the University of Ottawa. His current research work is on autonomous agents and multi-agent systems, including mobile trading agents.

Thomas Tran is an assistant professor in the school of information technology and engineering at the University of Ottawa. He received his PhD from the University of Waterloo in 2004. His research interests include intelligent agents, multi-agent systems, reinforcement learning, trust and reputation modeling, recommender systems and applications of AI to e-commerce.