Mobile Objects and Mobile Agents: The Future of Distributed Computing?[1]

Danny B. Lange

General Magic, Inc.
Sunnyvale, California, U.S.A
danny@acm.org, http://www.acm.org/~danny

Abstract. This paper will lead you into the world of mobile agents, an emerging technology that makes it very much easier to design, implement, and maintain distributed systems. You will find that mobile agents reduce the network traffic, provide an effective means of overcoming network latency, and perhaps most importantly, through their ability to operate asynchronously and autonomously of the process that created them, helps you to construct more robust and fault-tolerant. Read on and let us introduce you to software agents - the mobile as well as the stationary ones. We will explain all the benefits of mobile agents and demonstrate the impact they have on the design of distributed systems before concluding this paper with a brief overview of some contemporary mobile agent systems.

1 What's a Software Agent?

So what is a software agent? Well, what actually constitutes an agent, and how it differs from a normal program, has been heavily debated for several years now. While this debate is by no means over, we more and more often see agents loosely defined as *programs that assist people and act on their behalf.* This is what we prefer to call the "end-user perspective" of software agents.

Definition of an Agent (End-User Perspective)
An agent is a program that assists people and acts on their behalf. Agents function by allowing people to delegate work to them.

While this definition is basically correct, it does not really get under the hood. Agents come in myriad different types and in many settings. They can be found in computer operating systems, networks, databases, and so on. What properties do these agents share that constitute the essence of being an *agent*?

[1] This paper is based on a chapter of a book by Lange and Oshima entitled *Programming and Deploying Java™ Mobile Agents with Aglets*™, Addison-Wesley, 1998. (ISBN: 0-201-32582-9).

This is not the place to examine the characteristics of the numerous agent systems made available to the public by many research labs. But if you looked at all these systems, you would find that a property shared by all agents is that fact that they live in some environment. They have the ability to interact with their execution environment, and to act asynchronously and autonomously upon it. No one is required either to deliver information to the agent or to consume any of its output. The agent simply acts continuously in pursuit of its own goals.

In contrast to software objects of object-oriented programming, agents are active entities that work according to the so-called *Hollywood Principle*: "*Don't call us, we'll call you!*"

Definition of an Agent (System Perspective)

An agent is a software object that

- is situated within an execution environment;
- possesses the following mandatory properties:
    - Reactive - senses changes in the environment and acts accordingly to those changes;
    - Autonomous - has control over its own actions;
    - Goal driven - is pro-active;
    - Temporally continuous - is continuously executing;
- and may possess any of the following orthogonal properties:
    - Communicative - able to communicate with other agents;
    - Mobile - can travel from one host to another;
    - Learning - adapts in accordance with previous experience;
    - Believable - appears believable to the end-user.

2 What's a Mobile Agent?

Mobility is an orthogonal property of agents. That is, all agents do not necessarily *have* to be mobile. An agent can just sit there and communicate with the surroundings by conventional means. These include various forms of remote procedure calling and messaging. We call agents that do not or cannot move *stationary agents.*

Definition of a Stationary Agent

A stationary agent executes only on the system where it begins execution. If it needs information that is not on that system, or needs to interact with an agent on a different system, it typically uses a communication mechanism such as remote procedure calling (RPC).

In contrast, a *mobile* agent is not bound to the system where it begins execution. The mobile agent is free to travel among the hosts in the network. Created in one execution environment, it can transport its state and code with it to another execution environment in the network, where it resumes execution.

By the term "state," we typically understand the agent attribute values that help it determine what to do when it resumes execution at its destination. By the term "code," we understand, in an object-oriented context, the class code necessary for the agent to execute.

Definition of a Mobile Agent

A mobile agent is not bound to the system where it begins execution. It has the unique ability to transport itself from one system in a network to another. The ability to travel, allows a mobile agent to move to a system that contains an object with which the agent wants to interact, and then to take advantage of being in the same host or network as the object.

3 Seven Good Reasons for Using Mobile Agents

Although mobile agent technology sounds exciting, our interest in mobile agents should not be motivated by the technology *per se*, but rather by the benefits they provide for the creation of distributed systems. So here are seven good reasons for you to start using mobile agents.

They reduce the network load. Distributed systems often rely on communications protocols that involve multiple interactions to accomplish a given task. This is especially true when security measures are enabled. The result is a lot of network traffic. Mobile agents allow you to package a conversation and dispatch it to a destination host where the interactions can take place locally, see Figure 1. Mobile agents are also useful when it comes to reducing the flow of raw data in the network. When very large volumes of data are stored at remote hosts, these data should be processed in the locality of the data, rather that transferred over the network. The motto is simple: move the computations to the data rather than the data to the computations.

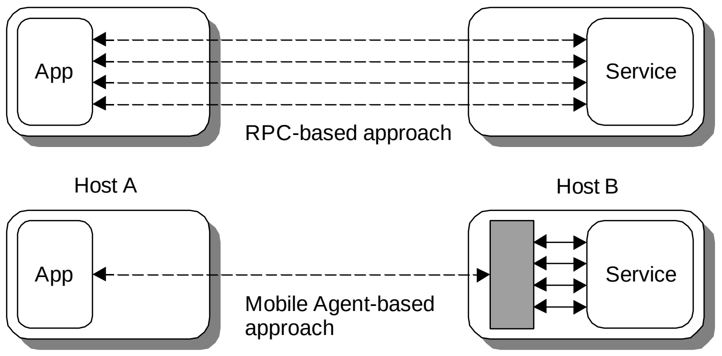

Fig. 1. Mobile Agents Reduce Network Load

They overcoming network latency. Critical real-time systems such as robots in manufacturing processes need to respond to changes in their environments in real time. Controlling such systems through a factory network of a substantial size involves significant latencies. For critical real-time systems, such latencies are not acceptable. Mobile agents offer a solution, since they can be dispatched from a central controller to act locally and directly execute the controller's directions.

They encapsulate protocols. When data are exchanged in a distributed system, each host owns the code that implements the protocols needed to properly code outgoing data and interpret incoming data, respectively. However, as protocols evolve to accommodate new efficiency or security requirements, it is a cumbersome if not impossible task to upgrade protocol code properly. The result is often that protocols become a legacy problem. Mobile agents, on the other hand, are able to move to remote hosts in order to establish "channels" based on proprietary protocols.

They execute asynchronously and autonomously. Often mobile devices have to rely on expensive or fragile network connections. That is, tasks that require a continuously open connection between a mobile device and a fixed network will most likely not be economically or technically feasible. Tasks can be embedded into mobile agents, which can then be dispatched into the network. After being dispatched, the mobile agents become independent of the creating process and can operate asynchronously and autonomously, see Figure 2. The mobile device can reconnect at some later time to collect the agent.

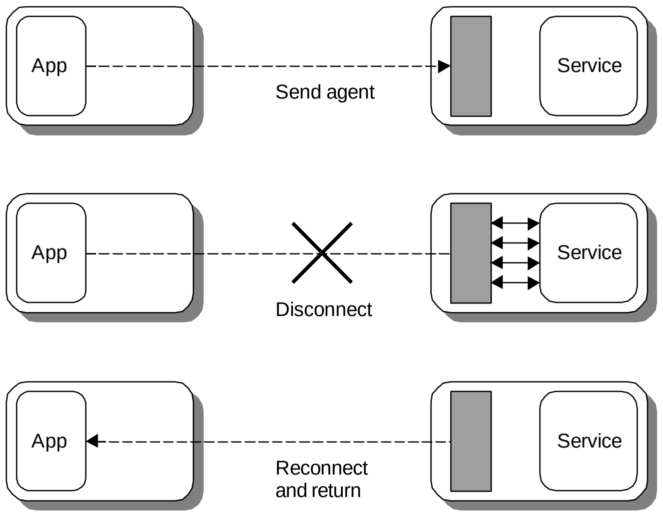

Fig. 2. Mobile Agents Allow Disconnected Operation

They adapt dynamically. Mobile agents have the ability to sense their execution environment and react autonomously to changes. Multiple mobile agents possess the unique ability to distribute themselves among the hosts in the network in such a way as to maintain the optimal configuration for solving a particular problem.

They are naturally heterogeneous. Network computing is fundamentally heterogeneous, often from both hardware and software perspectives. As mobile agents are generally computer- and transport-layer-independent, and dependent only on their execution environment, they provide optimal conditions for seamless system integration.

They are robust and fault-tolerant. The ability of mobile agents to react dynamically to unfavorable situations and events makes it easier to build robust and fault-tolerant distributed systems. If a host is being shut down, all agents executing on that machine will be warned and given time to dispatch and continue their operation on another host in the network.

4 Network Computing Paradigms

Our experience shows us that mobile agents provide a very powerful uniform paradigm for network computing. Mobile agents can revolutionize your design and development of distributed systems. To put this claim into perspective, we will provide a brief overview and comparison of three programming paradigms for distributed computing: *client-server*, *code-on-demand*, and *mobile agents*. Note that we put more emphasize on how the paradigm is perceived by the developer than on the underlying hardware-software architecture.

Client-Server Paradigm. In the client-server paradigm, see Figure 3, a server advertises a set of services that provide access to some *resources* (e.g., databases). The code that implements these services is hosted locally by the server. We say that the server holds the *know-how*. Finally, it is the server itself that executes the service, and thus has the *processor* capability. If the client is interested in accessing some resource hosted by the server, it will simply use one or more of the services provided by the server. Note that the client needs some "intelligence" to decide which of the services it should use. The server has it all, the know-how, resources, and processor. So far, most distributed systems have been based on this paradigm. We see it supported by a wide range of technologies such as remote procedure calling, object request brokers (CORBA), and Java remote method invocation (RMI).

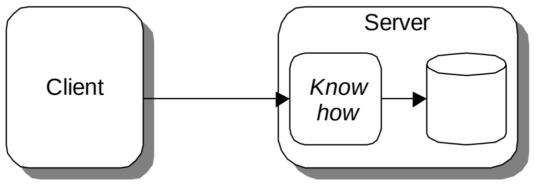

Fig. 3. Client-server Paradigm

Code-on-Demand Paradigm. Accordingly to the code-on-demand paradigm, see Figure 4, you first get the know-how when you need it. Say one host (A) initially is unable to execute its task due to a lack of code (know-how). Fortunately, another host

(B) in the network provides the needed code. Once the code is received by A, the computation is carried out in A. Host A holds the processor capability as well as the local resources. Unlike in the client-server paradigm, A does not need knowledge about the remote host, since *all* the necessary code will be downloaded. We say that one host (A) has the resources and processor, and another host (B) has the know-how. Java applets and servlets are excellent practical examples of this paradigm. Applets get downloaded in Web browsers and execute locally, while servlets get uploaded to remote Web servers and execute there.

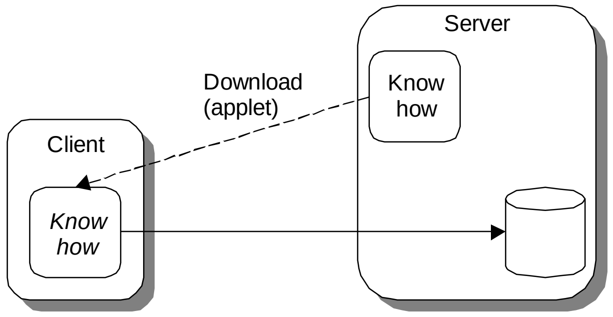

Fig. 4. Code-on-demand Paradigm

Mobile Agent Paradigm. A key characteristic of the mobile agent paradigm, see Figure 5, is that any host in the network is allowed a high degree of flexibility to possess any mixture of know-how, resources, and processors. Its processing capabilities can be combined with local resources. Know-how (in the form of mobile agents) is not tied to a single host but available throughout the network.

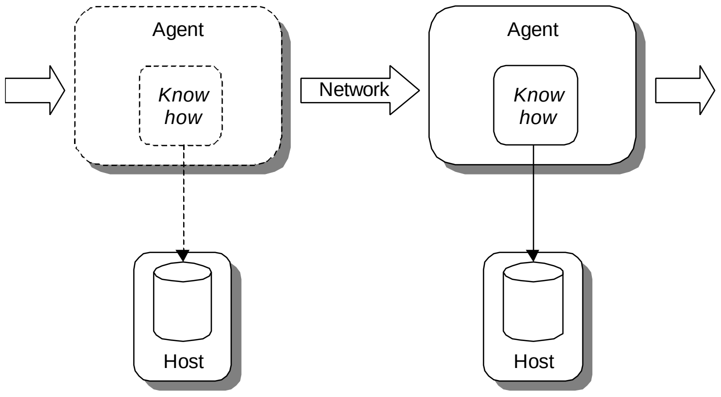

Fig. 5. Mobile Agent Paradigm

If you compare these three paradigms, you will see the chronological trend toward greater flexibility. The client and the server have merged and become a *host*. The applet and the servlet, while serving as client and server extenders, respectively, have been combined and improved with the emergence of mobile agents.

5 Mobile Agent Applications

We will now take a closer look at some applications that benefit particular from the mobile agent paradigm. Please note that this is by no means intended to be an exhaustive list.

Electronic commerce. Mobile agents are well suited for electronic commerce. A commercial transaction may require real-time access to remote resources such as stock quotes and perhaps even agent-to-agent negotiation. Different agents will have different goals, and will implement and exercise different strategies to accomplish these goals. We envision agents that embody the intentions of their creators, and act and negotiate on their behalf. Mobile agent technology is a very appealing solution to this kind of problem.

Personal assistance. The mobile agent's ability to execute on remote hosts makes it suitable as a "assistant" capable of performing tasks in the network on behalf of its creator. The remote assistant will operate independently of its limited network connectivity, and the creator can feel free to turn his or her computer off. To schedule a meeting with several other people, a user could send a mobile agent to interact with the representative agents of each of the people invited to the meeting. The agents could negotiate and establish a meeting time.

Secure brokering. An interesting application of mobile agents is in collaborations where not all the collaborators are trusted. In this case, the involved parties could let their mobile agents meet on a mutually agreed secure host, where collaboration can take place without the risk of the host taking the side of one of the visiting agents.

Distributed information retrieval. Information retrieval is an often-used example of a mobile agent application. Instead of moving large amounts of data to the search engine so that it can create search indexes, you dispatch agents to remote information sources, where they locally create search indexes that can later be shipped back to the origin. Mobile agents are also able to perform extended searches that are not constrained by the hours during which the creator's computer is operational.

Telecommunication networks services. Support and management of advanced telecommunication services are characterized by dynamic network reconfiguration and user customization. The physical size of these networks and the strict requirements under which they operate call for mobile agent technology to form the "glue" that keeps such systems flexible yet effective.

Workflow applications and groupware. It is in the nature of workflow to support the flow of information between co-workers. The mobile agent is particular useful here since, in addition to mobility, it provides a degree of autonomy to the workflow item. Individual workflow items fully embody the information and behavior needed for them to move through the organization independent of any particular application.

Monitoring and notification. This is one of the "classical" mobile agent applications that highlight the asynchronous nature of mobile agents. An agent is able to monitor a given information source without being dependent on the location from which it originates. Agents can be dispatched to wait for certain kinds of information to become available. It is often important that monitoring agents have life spans that exceed or are independent of the computing processes that create them.

Information dissemination. Mobile agents embody the so-called Internet "push" model. Agents are able to disseminate information such as news and automatic software updates for vendors. The agents will bring the new software components as well as the installation procedures directly to the customer's personal computer and will autonomously update and manage the software on the computer.

Parallel processing. Given that mobile agents can create a cascade of clones in the network, one potential use of mobile agent technology is to administer parallel processing tasks. If a computation requires so much processor power as to that it must be distributed among multiple processors, an infrastructure of mobile agent hosts could be a plausible way to get the processes out there.

6 Contemporary Mobile Agent Systems

So what kind of mobile agent systems are available for you? Fortunately, Java has generated a flood of experimental mobile agent systems. Numerous systems are currently under development, and most of them are available for evaluation on the Web.

The field is developing so dynamically and so fast that any attempt to map the agent systems will be outdated before this book goes to press. We will, however, mention a few interesting Java-based mobile agent systems: Aglets, Odyssey, Concordia, and Voyager.

Aglets. This system, created by the authors of this book, mirror the applet model in Java. The goal was to bring the flavor of mobility to the applet. The term *aglet* is indeed a portmanteau word combining *agent* and *applet*. We attempted to make Aglets an exercise in "clean design," and it is our hope that applet programmers will appreciate the many ways in which the aglet model reflects the applet model.

Odyssey. General Magic Inc. invented the mobile agent and created the first commercial mobile agent system called Telescript. Being based on a proprietary language and network architecture, Telescript had a short life. In response to the popularity of the Internet and later the steamrollering success of the Java language, General Magic decided to re-implement the mobile agent paradigm in its Java-based Odyssey. This system effectively implements the Telescript concepts in the shape of

Java classes. The result is a Java class library that enables developers to create their own mobile agent applications.

Concordia. Mitsubishi’s Concordia is a framework for the development and management of mobile agent applications which extend to any system supporting Java. Concordia consists of multiple components, all written in Java, which are combined together to provide a complete environment for distributed applications. A Concordia system, at its simplest, is made up of a standard Java VM, a Server, and a set of agents.

Voyager. ObjectSpace's Voyager is a platform for agent-enhanced distributed computing in Java. While Voyager provides an extensive set of object messaging capabilities it also allows object to move as agents in the network. You can say that Voyager combines the properties of a Java-based object request broker with those of a mobile agent system. In this way Voyager allows Java programmers to create network applications using both traditional and agent-enhanced distributed programming techniques.

We would like to note that the Java-based mobile agent systems have a lot in common. Beside the programming language they all rely on standard versions of the Java virtual machine and Java’s object serialization mechanism. A common server-based architecture permeates all the systems. Agent transport mechanisms and the support for interaction (messaging) varies a lot.

Although a majority of the contemporary mobile agent systems are based on the Java language system, you will also be able to find other languages in use. Most significant languages are Tcl and Python.

Agent Tcl. This is a mobile agent system whose agents can be written in Tcl. Dartmouth College’s Agent Tcl has extensive navigation and communication services, security mechanisms, and debugging and tracking tools. The main component of Agent Tcl is a server that runs on each machine and that allows the entire execution state including local variables and instruction pointer to move. When an agent wants to migrate to a new machine, it calls a single function, *agent_jump*, which automatically captures the complete state of the agent and sends this state information to the server on the destination machine. The destination server starts up a Tcl execution, loads the state information into this execution environment, and restarts the agent from the exact point at which it left off.

Ara. Tcl-based Ara from University of Kaiserslautern is a platform for the portable and secure execution of mobile agents in heterogeneous networks. The research project is primarily concerned with system support for general mobile agents regarding secure and portable execution, and much less with application-level features of agents, such as agent cooperation patterns, intelligent behavior, and user modeling.

TACOMA. The TACOMA project focuses on operating system support for agents and how agents can be used to solve problems traditionally addressed by operating systems. The TACOMA system is based on UNIX and TCP. The system supports agents written in C, Tcl/Tk, Perl, Python, and Scheme (Elk). The system itself is implemented in C.

Common for a number of the Tcl-based projects is that they anticipate a move toward support for multiple language which essentially means added support for Java.

We recommend the following Web sites for more information on the specific agent systems and projects:

- Aglets at `www.trl.ibm.co.jp/aglets`.
- Odyssey at `www.genmagic.com/agents`.
- Concordia at `www.meitca.com/HSL/Projects/Concordia`
- Voyager at `www.objectspace.com/voyager`.
- Agent Tcl at `www.cs.dartmouth.edu/~agent`.
- Ara at `www.uni-kl.de/AG-Nehmer/Ara`.
- TACOMA at `www.cs.uit.no/DOS/Tacoma`.

7 Mobile Agent Standardization: MASIF

Let us conclude this paper with a brief overview of ongoing standardization efforts in the mobile agent field.

Clearly, the above mentioned systems differ widely in architecture and implementation, thereby impeding interoperability and rapid deployment of mobile agent technology in the marketplace. To promote interoperability, some aspects of mobile agent technology must be standardized. The companies Crystaliz, General Magic Inc., GMD Fokus, IBM Corporation, and the Open Group have jointly developed a proposal for a *Mobile Agent System Interoperability Facility* (MASIF) and brought it to the attention of the Object Management Group (OMG).

MASIF addresses the interfaces between agent systems, not between agent applications and agent systems. Even though the former seems to be more relevant for application developers, it is the latter that allows mobile agents to travel across multiple hosts in an open environment. MASIF is clearly not about language interoperability. Language interoperability for mobile objects is very difficult and MASIF is limited to interoperability between agent systems written in the same language, but potentially by different vendors. Furthermore, MASIF does not attempt to standardize local agent operations such as agent interpretation, serialization, or execution. You can that say MASIF defines the interfaces at the agent system level rather than at the agent level.

MASIF standardizes the following four areas:

– **Agent Management.** There is interest in the mobile agent community to standardize agent management. It is clearly desirable that a system administrator who manages agent systems of different types can use the same standard operations. It should be possible to create an agent given a class name for the agent, suspend an agent's execution, resume its execution, or terminate it in a standard way.

– **Agent Transfer.** It is desirable that agent applications can spawn agents that can freely move among agent systems of different types, resulting in a common infrastructure.

– **Agent and Agent System Names.** In addition to standardizing operations for interoperability between agent systems, the syntax and semantics of various parameters must be standardized too. Specifically, agent name, and agent system name should be standardized. This allows agent systems and agents to identify each other, as well as applications to identify agents and agent systems.

– **Agent System Type and Location Syntax.** The location syntax must be standardized so that an agent can access agent system type information from a desired destination agent system. The agent transfer can only happen if the destination agent system type can support the agent. Location syntax also needs to be standardized so that agent systems can locate each other.

8 Summary

With agent *mobility* being the focus of this paper we defined a mobile agent as an agent that is not bound to the system where it begins execution. It has the unique ability to transport itself from one system in a network to another. The ability to travel, allows a mobile agent to move to a system that contains an object with which the agent wants to interact, and then to take advantage of being in the same host or network as the object. We gave you seven good reasons for you to start using mobile agents: they reduce the network load, they overcoming network latency, they encapsulate protocols, they execute asynchronously and autonomously, they adapt dynamically, they are naturally heterogeneous, and they are robust and fault-tolerant.

Among some of the application domains that benefits from mobile agent technology are: electronic commerce, personal assistance, secure brokering, distributed information retrieval, telecommunication networks services, workflow applications and groupware, monitoring and notification, information dissemination, and parallel processing.

Several Java-based mobile agent systems exist: Aglets, Odyssey, Concordia, Voyager, and many more. Although these Java-based mobile agent systems have a lot in common they do not interoperate. To promote interoperability, some aspects of mobile agent technology has been standardized by OMG's MASIF. It will be interesting to see what impact increased standardization activities will have on the mobile agent field.

References

1. Lange, D.B. and Oshima, M.: Programming and Deploying Java™ Mobile Agents with Aglets™, Addison-Wesley, 1998.
2. Aridor, Y. and Lange, D.B.: Agent Design Patterns: Elements of Agent Application Design, In Proceedings of the Second International Conference on Autonomous Agents (Agents '98), ACM Press, 1998, pp. 108-115.
3. Karjoth, G., Lange, D.B., and Oshima, M.: A Security Model for Aglets, IEEE Internet Computing 1, 4, 1997, pp. 68-77.
4. The Object Management Group: The Mobile Agent System Interoperability Facility, OMG TC Document orbos/97-10-05, The Object Management Group, Framingham, MA., 1997.
5. White, J.: Mobile Agents, In Software Agents, Bradshaw, J. Ed., MIT Press, 1997.