PART I
Introduction

1 Mobile Agents and Applications in Networking and Distributed Computing

JIANNONG CAO
Department of Computing, Hong Kong Polytechnic University

SAJAL K. DAS
Department of Computer Science and Engineering, The University of Texas at Arlington, USA

1.1 INTRODUCTION

Agent technology has evolved from two research areas: artificial intelligence and distributed computing. The purpose of AI research is to use intelligent computing entities to simplify human operations. An agent is just a computer program targeting that purpose [1]. Distributed computing, on the other hand, allows a complex task to be better executed by cooperation of several distributed agents on interconnected computers. So, networking and distribution bring out the true flavor of software agent technology in terms of agent autonomy, coordination, reactivity, heterogeneity, brokerage, and mobility.

Mobile agents refer to self-contained and identifiable computer programs that can move over the network and act on behalf of the user or another entity [2]. They can execute at a host for a while before halting the execution and migrating to another host and resuming execution there. They are able to detect the environment and adapt dynamically to changes. Mobile agents are widely used for handling disconnected operations in distributed, mobile, and wireless networking environments [3–6]. Also, many applications, including network diagnostic, e-commerce, entertainment and broadcasting, intrusion detection, and home health care, are benefited from the use of mobile agents [7, 8].

Mobile Agents in Networking and Distributed Computing, First Edition.
Edited by Jiannong Cao and Sajal K. Das.

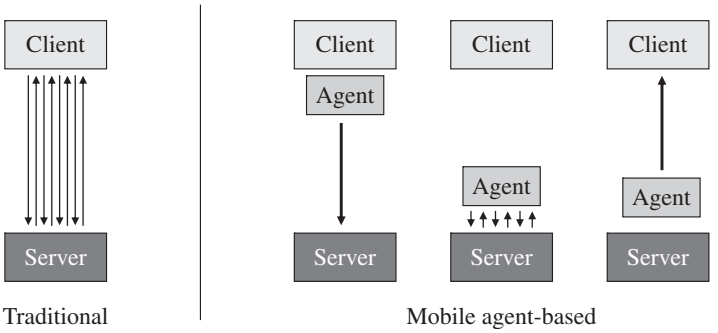

FIGURE 1.1 Mobile agent can reduce communication cost.

The term *mobile agent* contains two separate and distinct concepts: mobility and agency [9]. Some authors (e.g., [10]) classify a mobile agent as a special case of an agent, while others (e.g., [11]) separate the agency from mobility. Despite the differences in definition, most research on the mobile agent paradigm as reported in the literature has two general goals: reduction of network traffic and asynchronous interaction. Mobile agents can reduce the connecting time and bandwidth consumption by processing the data at the source and sending only the relevant results. By moving the agents to data-residing hosts, they can reduce communication costs. On the other hand, mobile agents support asynchronous interaction. They can continue computations even if the user that has started it, is no longer connected to the system. Mobile agents have been proposed as an alternative to the client–server paradigm which can be a more efficient and flexible mode of communication in certain application areas (Figure 1.1). It has been recognized that mobile agents provide a promising approach to dealing with dynamic, heterogeneous, and changing environments, which is tendency of modern Internet applications.

A mobile agent has the following properties or capabilities [12, 13]:

Mobility Transport itself from host to host within a network. This is the most distinguishing property from other kinds of agents. Note that a moving agent will carry its identity, execution state, and program code so that it can be authenticated and hence can resume its execution on the destination site after the move. Mobility refers to a wide range of new concepts. *Migration* is undoubtedly the most important of these concepts. Migration allows an agent to move from one location to another. The migration of a mobile agent requires the agent system to support execution stopping, state collection, data serialization and transfer, data deserialization, and execution resuming. From this point of view, mobile agents strongly rely on mobile code technology, which will be described in detail later in this chapter.

Intelligence Interact with and learn from the environment and make decisions. A most advanced agent should be able to decide its action

based on its knowledge and the information it gets en route, and thus be able to generate new knowledge from its experience.

Autonomy Take control over its own actions. An agent should be able to execute, move, and settle down independently without supervision even in long-term running.

Recursion Create child agents for subtasks if necessary. An important concept is agent *cloning*: The agent can clone itself, that is, create a new mobile agent that is a copy of the parent. A pure cloning operation implies that the cloned agent has the same behavior (code) and the same knowledge (data) as the parent agent. A postcloning operation can initialize specific values in the cloned agent, which starts its life cycle in the same execution environment as the parent. Its location can however be different from the parent's.

Asynchrony In a distributed computing environment, perform computation concurrently and possibly on different sites. Also, performing computation on behalf of its user, an agent is responsible for the task assigned by a user and allows the user to offer and/or obtain resources and services in order to finish the task. All these can be done asynchronously with the user's action.

Collaboration Cooperate and negotiate with other agents. Complicated tasks can be carried out by collaboration of a group of agents.

1.2 MOBILE AGENT PLATFORMS

A mobile agent platform (MAP) is a software package for the development and management of mobile agents. It is a distributed abstraction layer that provides the concepts and mechanisms for mobility and communication on the one hand, and security of the underlying system on the other hand. The platform gives the user all the basic tools needed for creating some applications based on the use of agents. It enables us to create, run, suspend, resume, deactivate, or reactivate local agents, to stop their execution, to make them communicate with each other, and to migrate them.

Some agent standards enable interoperability between agent platforms so that software agents can communicate and achieve their objectives according to standardized specifications. The most popular agent standards are *FIPA* and *OMG-MASIF* as discussed below.

1.2.1 FIPA

The *Foundation for Intelligent Physical Agents* (FIPA) was formed in 1996 to produce software standards for heterogeneous interacting agents and agent-based systems. Currently, FIPA appears to be the dominant standards organization in the area of agent technology. Important efforts have been made to address the interoperability issues between the agent platforms. Figure 1.2

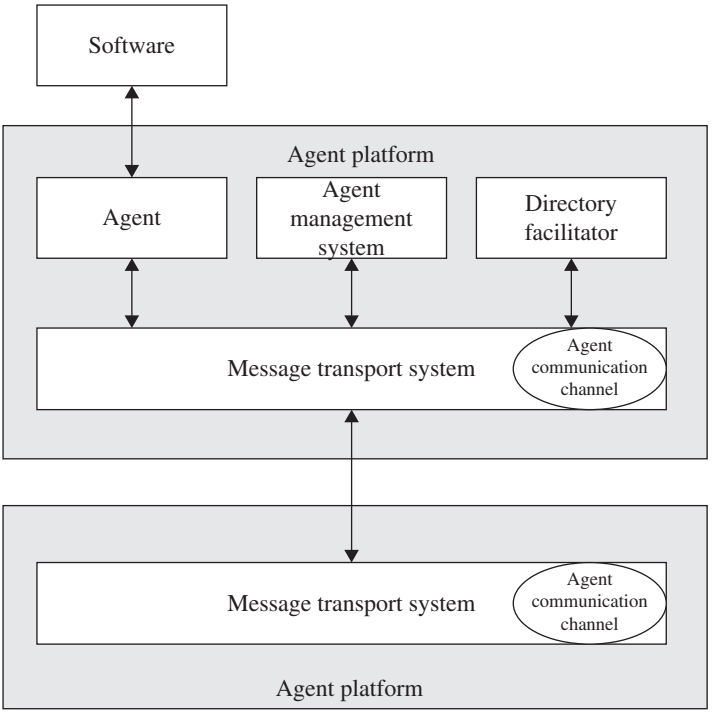

FIGURE 1.2 Agent system reference model of FIPA.

presents the overall architecture of an agent system as specified by FIPA. The message transport is the main underlying mechanism devoted to communication between agents based on Agent Communication Language (ACL); at this stage, mobile agents are not supported. The message transport itself relies on standard communication techniques used by distributed system frameworks, such as Common Object Request Broker Architecture (CORBA) or *Java* remote method invocation (RMI).

Both *Agent Management System* (AMS) and *Directory Facilitator* (DF) are FIPA agents: the AMS is responsible for the core management activities of the agent platform whereas the DF acts as a *yellow page* service. Agents are registered in the DF and can be localized from their types by other agents. In addition, agent communication is ensured through the *Message Transport System* (MTS), including the *Message Transport Protocol* (MTP) and the *Agent Communication Channel* (ACC), which directly provide agents with specific services for communication. The ACC may access information provided by the other agent platform services such as the AMS and DF to carry out its message transport tasks.

1.2.2 OMG-MASIF

In 1997, the Object Management Group (OMG) released a draft version of the *Mobile Agent System Interoperability Facilities* (MASIF) [14]. MASIF

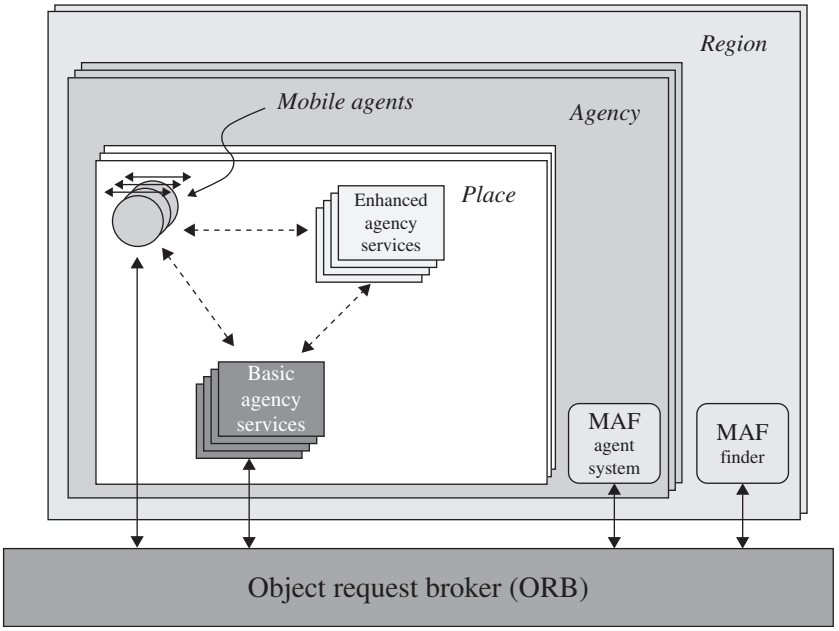

FIGURE 1.3 General architecture of OMG-MASIF mobile agent system.

proposes a specification of the communication infrastructure as well as interfaces defined in an interface definition language (IDL) to access mobility services in order to promote the interoperability and diversity of MAP. From the interoperability and heterogeneity perspectives, OMG follows the same objectives as FIPA. The objectives in terms of requirements and functionalities are clearly different, however. Whereas FIPA is concerned with a message-based communication infrastructure, MASIF has to take into account the migration of the agent and must consequently focus on the way to dynamically *create the agent*, that is, to instantiate a new object at the right place and with the right class.

In Figure 1.3, the MASIF architecture appears to be a hierarchical organization of regions, agencies, and places [15]. The *place* is a context within an agent system in which an agent can execute its tasks and provide local access control to mobile agents. A place is associated with a location which consists of the place name and the address of the agent system within which the place resides. The *agency* represents the agent system itself or is the core part of the agent system. At a higher level, the *region* is a set of agent systems that have the same authority but are not necessarily of the same type.

Considering its origin, MASIF strongly relies on a CORBA architecture and therefore on the ORB. The services provided by the region, agency, and place are defined through IDL interfaces; the most important interfaces are MAFFinder and MAFAgentSystem: the MAFFinder supports the localization of agents, agent systems and places in the scope of a region or in the whole

environment; on the other hand, the MAFAgentSystem interface provides operations for the management and transfer of agents. In MASIF, the agent's migration requires the transfer of the agent class so that the agent can be properly instantiated.

1.3 REPRESENTATIVE MAPs

In the following, we briefly describe some representative MAPs.

1.3.1 IBM Aglets Workbench (1997–2001)

This is a Java MAP and library that eases the development of agent-based applications. Originally developed at the IBM Tokyo Research Laboratory, the Aglets technology is now hosted at sourceforge.net as an open-source project, where it is distributed under the IBM Public License. Aglets is completely made in Java, granting a high portability of both the agents and the platform. The aglet represents the next leap forward in the evolution of executable content on the Internet, introducing program code that can be transported along with state information. Aglets are Java objects that can move from one host on the Internet to another. That is, an aglet that executes on one host can suddenly halt execution, dispatch itself to a remote host, and resume execution there. When the aglet moves, it takes along its program code as well as its data.

1.3.2 Agent Tcl (1994–2002, later known as D'Agents)

This does not formally specify a mobile agent model. Instead, a mobile agent is understood as a program that can be written in any language and that accesses features that support mobility via a common service package implemented as a server. This server provides mobile agent–specific services such as state capture, transfer facility, and group communication as well as more traditional services such as disk access, screen access, and CPU cycle. The philosophy was that all functionalities an agent ever wants are available in the server. Agent mobility then only concerns closure, which is the Tcl script (or scripts). There are no additional codes to load (i.e., no external references). In Agent Tcl, the state capture of an agent is handled automatically and transparently to the programmer. However, it is unclear what this state capture includes. Since Tcl is a script language, a frequent example given is that the executing script resumes after the instruction for mobility has been executed. There is also a plan to introduce process migration–like behavior such that the states of the agent would continue to evolve as it moves from place to place. However, this trend could have adverse effects in areas such as the complexity of the transfer mechanism and cost, adverse effects that are still being dealt with in the more traditional process migration.

1.3.3 Grasshopper (1998)

The agent development platform launched by IKV++ in August 1998, enables the user to create a wealth of applications based on agent technology. This platform is completely implemented in Java, a programming language that has become widely known among programmers, giving them the opportunity to work with Grasshopper without intensive further training. Companies with an urgent need for true distributed systems can therefore benefit almost immediately from the advantages of Java as well as from Grasshopper's unique suitability for such systems. Grasshopper is also the first mobile agent environment that is compliant to the industry standard supporting agent mobility and management (OMG-MASIF). This compliance ensures compatibility with other agent environments or applications based on the same standard, thus avoiding costly and time-consuming integration procedures. From Grasshopper version 1.2 released in 1999, it is also compliant with the specifications of the FIPA standards. Grasshopper can be used in many different application contexts, telecommunications being one of the most prominent application areas.

1.3.4 Concordia (1997)

This is another mobile agent framework built on Java. In Concordia an agent is regarded as a collection of Java objects. A Concordia agent is modeled as a Java program that uses services provided by a collection of server components that would take care of mobility, persistence, security, communication, administration, and resources. These server components would communicate among themselves and can run in one or several Java virtual machines; the collection of these components forms the agent execution environment (AEE) at a given network node. Once arriving at a node, the Concordia agent accesses regular services available to all Java-based programs such as database access, file system, and graphics, as in Aglet. A Concordia agent is considered to have internal states as well as external task states. The internal states are values of the objects' variables, while the external task states are the states of an itinerary object that would be kept external to the agent's code. This itinerary object encapsulates the destination addresses of each Concordia agent and the method that each would have to execute when arriving there. The designers of Concordia claim that this approach allows greater flexibility by offering multiple points of entry to agent execution, as compared to always executing an "after-move" method as in Agent Tcl, or Aglet. This concept of an externally located itinerary is similarly supported in Odyssey via task object. However, the infrastructure for management of these itinerary objects is not clear from the publicly available literature on Concordia which has support for transactional multiagent applications and knowledge discovery for collaborating agents.

1.3.5 In Mole (1997)

The agent is modeled as a cluster of Java objects, a closure without external references except with the host system. The agent is thus a transitive closure over all the objects to which the main agent object contains a reference. This island concept was chosen by the designers of Mole to allow simple transfer of agents without worrying about dangling references. Each Mole agent has a unique name provided by the agent system which is used to identify the agent. Also, a Mole agent can only communicate with other agents via defined communication mechanisms which offer the ability to use different agent programming languages to convert the information transparently when needed. A Mole agent can only exist in a host environment call *location* that serves as the intermediate layer between the agent and the operating system. Mole also supports the concept of *abstract location* to represent the collection of distributed physical machines. One machine can contain several locations, and locations may be moved among machines. Mole limits the abstract location to denote a configuration that would minimize cost due to communication. Thus, a collection of machines in a subnet is an acceptable abstract location, whereas a collection of machines that spans cities is not. Mole proposed the concept of a system agent which has full access to the host facilities. It is through interacting with these system agents that a given Mole agent (mobile) achieves tasks. A Mole mobile agent can only communicate with other agents (systems and mobile agents) and has no direct access to resources. The uniqueness of this agent model is its requirement for closure of objects, whereas other facilities such as static agent and communication are conceptually similar to other systems. What is unclear is how the Mole system enforces the closure requirement and whether there are mechanisms to handle closure management automatically. The concept of closure is technically convenient, but without helping tools it can be error prone and thus limiting.

1.3.6 The Odyssey

Project shares (or rather inherits) many features from a previous General Magic product: Telescript. However, the amount of open documentation on the Odyssey system is rather terse; therefore, its description is limited. The Odyssey mobile agent model also centers on a collection of Java objects, more similar in concept to Aglet than to Concordia or Mole. The top-level classes of the Odyssey system are *Agent*, *Worker*, and *Place*. Worker is a subclass of Agent and represents an example of what a developer can do with the Agent class. A Place class is an abstraction of where an Odyssey agent exists and performs work. A special facility such as directory service is associated with Place. Odyssey agents communicate using simple method calls, and do not support high-level communication. However, Odyssey agents can form and destroy meeting places to exchange messages. There is also an undocumented feature regarding global communication to a "published" object, but this

feature is not officially supported. The distinctive feature of Odyssey is its design to accommodate multiple transport mechanisms. Currently, Odyssey supports Java RMI, Microsoft Distributed Component Object Model (DCOM), and CORBA Internet Inter-ORB Protocol (IIOP). However, the current release of Odyssey does not add new or distinctive features from its Telescript predecessor, and the mobile agent model is not yet stable.

1.4 SOME APPLICATIONS

All the above mobile agent platforms are targeted at providing execution environment and programming support for developing applications. Primitive language-level operations required by programmers for developing agent-based applications are identified. They are (1) *basic agent management functions*, such as creation, dispatching, cloning, and migration; (2) *agent-to-agent communication and synchronization functions*; (3) *agent monitoring and control functions*, such as status queries, recall, and termination of agents; (4) *fault tolerance functions,* such as check pointing, exception handling, and audit trails; and (5) *security-related functions*, such as encryption, authentication, signing, and data sealing.

As mentioned before, many mobile agent–based applications have been studied. Readers can find surveys on various types of applications [7, 8]. Based on earlier mobile agent platforms, several new platforms have been developed to meet the requirements of newly emerging computing technologies and applications, including mobile computing, ad hoc networking, and ubiquitous or pervasive computing [6, 16–20].

For distributed and network computing, mobile agent technology has been used to design both system functions and applications. This book includes excellent tutorial and advanced materials that cover a wide range of topics. Here, we just describe one of the typical mobile agent applications that can help reduce network communication cost. The mobile agent is particularly attractive as a promising technology for information retrieval in large-scale distributed systems like the Internet. The mobile agent acts as task-specific executable code traveling the relevant information source nodes to retrieve data. Several approaches have been proposed with both experimental and analytical evaluations [21–23].

More recently, the mobile agent has been used in designing dynamic and ad hoc systems. It enables the system to have the ability to deal with the uncertainty in a dynamic environment. For example, works have been reported [24, 25] on using mobile agents for monitoring, traffic detection, and management in highly dynamic distributed systems. Other examples include using mobile agents for ad hoc networks [6, 26].

Mobile agents are also being used for developing applications for wireless sensor networks (WSNs). Various operations and system functions in WSNs can be designed and implemented using mobile agents, which can greatly

reduce the communication cost, especially over low-bandwidth links. Efficient data dissemination and data fusion in sensor networks using mobile agents have been proposed [4, 27, 28]. Solutions to location tracking in sensor networks using mobile agents are also proposed [29]. Location tacking aims to monitor the roaming path of a moving object. There are two primary challenges: no central control mechanism and backbone network in such environment and the very limited wireless communication bandwidth. A mobile agent can assist in tracking such a mobile object by choosing to migrate in the sensor closest to the object. For programming support, mobile agent–based WSN middleware has been developed as a better foundation for rapidly developing flexible applications for WSNs [5, 30]. Also WSN-based structural health monitoring applications use mobile agent–based network middleware [31] to enhance flexibility and to reduce raw data transmission. Design of wireless sensor networks for structural health monitoring presents a number of challenges, such as adaptability and the limited communication bandwidth. In [31], an integrated wireless sensor network consisting of a mobile agent–based network middleware and distributed high computational power sensor nodes has been developed. The mobile agent middleware is built on a mobile agent system called Mobile-C that allows a sensor network to move computational programs to the data source. With mobile agent middleware, a sensor network is able to adopt newly developed diagnosis algorithms and make adjustments in response to operational or task changes.

1.5 OVERVIEW OF THE BOOK

As briefly described in the previous sections, there exists many applications to benefit from mobile agent technology such as e-commerce, information retrieval, process coordination, mobile computing, personal assistance, and network management. Still more and more applications are switching to use mobile agents due to their flexibility and adaptability. Also their abilities of asynchronous and autonomous execution make connectionless execution possible, which might be extremely valuable in the mobile computing context. By moving computation to data rather than data to computation, mobile agents can also reduce the flow of raw data in the network and therefore overcome network latency, which is especially critical to real-time applications. Additionally, other distinguishable features, such as fault tolerance, natural heterogeneity, and protocol encapsulation enhance the utilization and application horizon of the mobile agent technology over traditional approaches.

This book focuses on cutting-edge research and applications of mobile agent technology in the areas of networking and distributed computing. The book is divided into four parts: (1) introduction, (2) principles of applying mobile agents to networking and distributed computing, (3) mobile agents techniques as applied to networking and distributed computing, and (4) design and evaluation.

The first part introduces the idea of mobile agents and discusses their potential as an important tool in networking and distributed computing.

The second part will show how to apply mobile agents to networking and distributed computing. In this part, we cover mobile agent communication, coordination, and cooperations as well as mobile agent security mechanisms. Agents must communicate with each other in order to solve problems together. *Communication* has been viewed between agents as planned actions that are not aimed at changing the environment; rather the aim is to change the beliefs and intentions of the agent to whom the message is sent. This implies that social agents should have a framework with which to analyze each other's behavior. Mobile agent *coordination* is mainly required for distributed programs consisting of a team of cooperating agents, where each agent is responsible for performing part of a common, global task. Teams of mobile agents are likely to become the means to implement several distributed and networked applications in the future. For example, one possible application is the search for some information in the network to be performed in parallel by a group of agents that will not visit the same host more than once. *Cooperation* between a collection of mobile agents is required for exchanging information or for engaging in cooperative task-oriented behaviors. In addition to the advantages of the mobile agent, using a cooperating mobile agent allows us to provide clear and useful abstractions in building network services through the separation of different concerns. Furthermore, mobile agents can be used to perform intrusion detection. With mobile agent technology, the collection nodes, internal aggregation nodes, and command and control nodes do not have to continuously reside on the same physical machine. For example, a mobile agent may function as an aggregation node and move to whatever physical location in the network is best for its purposes.

The third part of the book, describes in detail the techniques of mobile agents in networking. Especially we discuss the applications of agents in network routing, resource and service discovery, distributed control, distributed databases and transaction processing, and wireless and mobile computing. Mobile agents can have interesting applications at the network infrastructure layer. Agents can adapt the network infrastructure to changing needs over time and can facilitate network routing. Mobile agents can also dynamically discover resources they need to accomplish their tasks. When an agent arrives at a site, it should be able to discover the services offered at that site or things it could do. Distributed control using mobile agents is a useful approach for load balancing, deadlock detection, mutual exclusion, and so on. Characteristics of mobile agents make them useful in achieving load balance in the whole system. We also present some distributed algorithms using mobile agent systems for mutual exclusion, deadlock detection, consensus, and so on. Using transactions for managing large data collections will guarantee the consistency of data records when multiple users or processes perform concurrent operations on them. Owing to the heterogeneous and autonomous environment that the mobile agents operate in and their typical longevity, agent-based

transactions have specific requirements. We discuss those requirements and possible recovery mechanisms. With the advent of mobile wireless communications and the growth of mobile computing devices, such as laptop computers, personal digital assistants (PDAs), and cell/smart phones, there is a growing demand for mobile agent–based mobile computing middleware and the mobile agent platforms for wireless hand-held devices and pervasive computing [32].

In the final part, we will discuss the means of measuring performances of mobile agent systems during the development of agent code, such as capturing the overhead of local agent creation, point-to-point messaging, and overhead for agent roaming. We can a keep track of the execution-related performances of mobile agents, such as the migration performance.

REFERENCES

1. A. Lingnau, O. Drobnik, and P. Domel, An HTTP-based infrastructure for mobile agents, *WWW J., Proceedings 4th International WWW Conference*, Vol. 1, Dec. 1995, pp. 461–471.
2. K. Rothermel and R. Popescu-Zeletin, Eds., Mobile agents, Lecture Notes in Computer Science, 1219, Springer, 1997.
3. J. Cao, Y. Sun, X. Wang, and S. K. Das, Scalable load balancing on distributed web servers using mobile agents, *J. Parallel Distrib. Comput.*, 63(10):996–1005, Oct. 2003.
4. M. Chen, S. Gonzalez, and V. C. M. Leung, Applications and design issues for mobile agents in wireless sensor network, *IEEE Wireless Commun.*, 14(6):20–26, Dec. 2007.
5. C.-L. Fok, G.-C. Roman, and C. Lu, Agilla: a mobile agent middleware for self-adaptive wireless sensor networks, *ACM Trans. Auton. Adapt. Syst.*, 4(3), July 2009.
6. J. Park, H. Yong, and E. Lee, A mobile agent platform for supporting ad hoc network environment, *Int. J. Grid Distrib. Comput.*, 1(1), 2008.
7. D. Milojicic, Mobile agent applications, *IEEE Concurrency*, July–Sept. 1999.
8. A. Outtagarts, Mobile agent-based applications: A survey, *IJCSNS Int. J. Comput. Sci. Network Security*, 9(11), Nov. 2009.
9. V. A. Pham and A. Karmouch, Mobile software agents: An overview, *IEEE Commun. Mag.*, 36(7):26–37, July 1998.
10. H. S. Nwana and N. Azarmi, Eds., Software agents and soft computing: Towards enhancing machine intelligence, Lecture Notes AI Series, 1198, Springer, 1997.
11. J. Vitek and C. Tschudin, Eds., Mobile object systems: Towards the programmable internet, Lecture Notes in Computer Science, 1222, Springer, 1997.
12. J. White, Prospectus for an open simple agent transfer protocol. White paper, General Magic, online, 1996.
13. A. Piszcz, A brief overview of software agent technology. White paper, The MITRE Corporation, McLean, VA, 1998.

14. GMD Fokus, Mobile agent system interoperability facilities specification, OMG TC Document orbos/97-10-05, Nov. 1997. (OMG homepage - www.omg.org)
15. C. Baumer, M. Breugst, S. Choy, and T. Magedanz, Grasshoper: a universal agent platform based on OMG MASIF and FIPA Standards, http://www.ikv.de/products/grasshopper.html.
16. J. Cao, D. C. K. Tse, A. T. S. Chan, PDAgent: A platform for developing and deploying mobile agent-enabled applications for wireless devices, in *Proceedings of 2004 International Conference on Parallel Processing (ICPP'2004)*, Montreal, Quebec, Canada, Aug. 2004, pp. 510–517.
17. F. Bagci, J. Petzold, W. Trumler, and T. Ungerer, Ubiquitous mobile agent system in a P2P-Network, paper presented at the UbiSys-Workshop at the Fifth Annual Conference on Ubiquitous Computing, Seattle, WA, Oct. 12–15, 2003.
18. M. Kumar, B. Shirazi, S. K. Das, B. Sung, D. Levine, and M. Singhal, PICO: A middleware framework for pervasive computing, *IEEE Pervasive Comput.*, 2(3):72–79, July–Sept. 2003.
19. J. R. Kim and J. D. Huh, Context-aware services platform supporting mobile agents for ubiquitous home network, in *Proceedings of the 8th International Conference on Advanced Communication Technology (ICACT'2006)*, Phoenix Park, Gangwon-Do, Korea, February 20–22, 2006.
20. G. S. Kim, J. Kim, H.-j. Cho, W.-t. Lim, and Y. I. Eom, Development of a lightweight middleware technologies supporting mobile agents, Lecture Notes in Computer Science, Vol. 4078, 2009.
21. S. Pears, J. Xu, C. Boldyreff, Mobile agent fault tolerance for information retrieval applications: An exception handling approach, *in Proceedings of the 6th International Symposium on Autonomous Decentralized Systems (ISADS'03)*, 2003.
22. W. Qu, M. Kitsaregawa, and K. Li, Performance analysis on mobile-agent based parallel information retrieval approaches, *in Proceedings of 2007 IEEE International Conference on Parallel and Distributed Systems*, Dec. 5–7, 2007.
23. W. Qu, W. Zhou, and M. Kitsaregawa, An parallel information retrieval method for e-commerce, *Int. J. Comut. Syst. Sci. Eng.*, 5:29–37, 2009.
24. B. Chen, H. H. Cheng, and J. Pelen, Integrating mobile agent technology with multi-agent systems for distributed traffic detection and management, *Transport. Res. Part-C.*, 17, 2009.
25. J. Ahn, Fault tolerant mobile-agent based monitoring mechanism for highly dynamic distributed networks, *Int. J. Comput. Sci. Iss.*, 7(3):1–7, May 2010.
26. G. Stoian, Improvement of handoff in mobile WiMAX network using mobile agent, *in Latest Trends in Computers*, Vol. 1, WSEAS Press, 2010, pp. 300–305.
27. Q. Hairong, S. Iyengar, and K. Chakrabarty, Multiresolution data integration using mobile agents in distributed sensor networks, *IEEE Trans. Syst. Man Cybernet.*, 31(3):383–391, August 2001.
28. Q. Wu, N. S. V. Rao, and J. Barhen, On computing mobile agent routes for data fusion in distributed sensor networks, *IEEE Trans. Knowledge Data Eng.*, 16(6):740–753, June 2004.

29. Y.-C. Tseng, S.-P. Kuo, H.-W. Lee, and C.-F. Huang, Location tracking in a wireless sensor network by mobile agents and its data fusion strategies, *Comput. J.*, 47(4):448–460, July 2004.

30. C.-L. Fok, G.-C. Roman, and C. Lu, Mobile agent middleware for sensor networks: An application case study, *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (IPSN)*, Los Angeles, CA, 2005, pp. 382–287.

31. B. Chen and W. Liu, Mobile agent computing paradigm for building a flexible structural health monitoring sensor network, *Comput.-Aided Civil Infrastruct. Eng.*, 25(7):504–516, October 2010.

32. Y. Feng, J. Cao, I. Lau, Z. Ming, and J. Kee-Yin Ng, A component-level self-configuring personal agent platform for pervasive computing, *Int. J. Parallel, Emergent Distrib. Syst.* 26(3):223–238, June 2011.