ISSN No. 0976-5697

Volume 4, No. 4, March-April 2013

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

A Framework for Accessing Remote Machine Configuration using Mobile Agents

Pravada S. Bharatkar
M. Tech. Scholar
Dept. of Computer Technology,
Rajiv Gandhi College of Engineering
Research and Technology, Chandrapur
pravadadilip@gmail.com

Abstract: Mobile agents (MA) systems have gained popularity in use because they ease the application design process by giving software engineers greater flexibility. The research in MA is a rapidly growing field contributing to autonomous software agents and distributed systems. And related is affecting the world of network computing and the agents technology is well suited for the network application. Mobility is both a useful abstraction and tool for agent-based system designers; it allows for increased resource efficiency, capability, and robustness. It is expected the mobile agent to be mobile and be able to do collaboration, integrating these technologies, to facilitated network user in retrieving information. For fulfilling these aspects, the frame for accessing remote machine configuration using MA is developed. Therefore, in the present study, the various non-java and java based mobile agent systems are studied for their strategic application in the area of integration and gathering of the information. The InfoGatherAgent java program is designed and implemented for gathering the information from the different ports of Aglets-Tahiti Server to reduce the network load.

Key words-mobile agents; aglets-tahiti server; infogatheragent; information gathering

I. INTRODUCTION

In the present era of technology, computers are fulfilling an increasingly diverse set of tasks in our society. They are providing seamless assistance to support our lifestyles through silently assuming many mundane but key tasks, e.g. they control our car engines, our environmental climate and even our toasters, etc. Increasingly sophisticated hardware is the supporting substrate for increasingly complex software. Yet despite major advances in our understanding of the construction of software, building flexible and reliable systems remains a considerable task. Intensifying powerful abstractions are employed by software engineers in an attempt to reduce the cognitive complexity of such tasks. Over the years computer systems have evolved from centralized monolithic computing devices supporting static applications, into client-server environments that allow complex forms of distributed computing. Throughout this evolution limited forms of code mobility have existed. The earliest being remote job entry terminals used to submit programs to a central computer and the latest being Java applets downloaded from web servers into web browsers. A new phase of evolution is now under way that goes one step further, allowing complete mobility of cooperating applications among supporting platforms to form a large-scale, loosely-coupled distributed system. The catalysts for this evolutionary path are mobile software agents-programs that are goal-directed and capable of suspending their execution on one platform and moving to another platform where they resume execution.

A. Mobile Agents:

Mobile agents (MA) are autonomous software agents that travel in a computer network to execute and perform tasks on different hosts on behalf of their owners. Autonomous mobile agents bring advantages such as task delegation, network communication, and cost reduction for distributed tasks [1]. MA systems provide a great flexibility and customizability to distributed applications like e-business and information retrieval in the current scenario. The technology of MA offers a new computing paradigm in which a program, in the form of a software agent, can suspend its execution on a host computer, transfer itself to another agent-enabled host on the network, and resume execution on the new host. The use of mobile code has a long history dating back to the use of remote job entry systems in the 1960's. Today's agent incarnations can be characterized in a number of ways ranging from simple distributed objects to highly organized software with embedded intelligence.

As such, mobile agents are processes (e.g. executing programs) that can migrate from one machine of a system to another machine (usually in the same system) in order to satisfy requests made by their clients [2]. It is a software program with mobility which can be sent out from a computer into a network and roam among the computer nodes in the network [3]. The key characteristic of the mobile agent paradigm 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 [4]. MA has the unique ability to transport itself from one system in a network to another in the same network. This ability allows it to move to a system containing an object with which it wants to interact and then to take advantage of being in the same host or network as the object. The huge information regarding the use of mobile agent is found in literature [5, 6, and 7]. The seven good reasons for use of mobile agent [8] are described by Danny and Mitsuru. Several academic and industrial research groups are currently investigating and building mobile agent systems. Generally, the Non-Java and Java based mobile agent system are existed in literatures [9]. The examples of Non-Java mobile agent systems are Telescript [10], Agent Tcl [11], D'Agents, Ara [12], TACOMA [13], while Java-based mobile agent systems are

Aglets [14], Concordia [15], NOMADS. The representative subset of this mobile agent system is described as follows:

a. Telescript :

Telescript is developed by General Magic in 1990's as the first system designed expressively to support mobile agent in commercial application as well as object oriented agent programming. It extensively supports security and access control.

b. Tacoma:

Tacoma is a joint project of Norway's University of Tromso and Cornell Univeristy. It uses check pointing and provides rearguard agent for tracking mobile agents as they migrate.

c. Agent Tcl:

Agent Tcl developed at Dartmouth and allows Tcl script to migrate between servers that support agent execution, communication, status queries, and non-volatile storage.

d. Java:

Although not marketed as a mobile agent framework, the Java [16] Development Kit does provide enough native facilities to support weakly mobile code. The most widely known examples of Java's mobile code capabilities are probably applets and servlets.

e. D'Agents:

Developed at Dartmouth College, D'Agents is one of the new breeds of mobile agent framework.

f. Mole:

Mole [17] was the first mobile agent framework developed in Java, and was initially released in 1995 by the IPVR group of Stuttgart University.

g. Hive:

Hive is a distributed agents platform, a decentralized system for building applications by networking local system resources, and taking advantage of mobile code [18]. Hive is built using the standard Java features of object serialization and interpretation used by so many mobile agent frameworks.

h. Voyager:

Voyager is Java based agent system developed by Object Space features. At the time of writing Voyager currently supports EJB [19], CORBA, DCOM, and RMI.

i. Jini:

Jini [20] is Sun Microsystem's proposed architecture for embedded network applications. It is built using Java and RMI in much the same way as Hive.

j. Aglets:

The Aglet Software Development Kit (ASDK) [21] has been developed by IBM's Tokyo Research Labs, and was one of the first and most publicized Java based mobile agent frameworks released. The core abstractions supported by the ASDK are that of an aglet, a proxy and a context.

The comparison among three Mobile Agent Systems [22] such as Aglets, Grasshopper and Voyager, this shows that the Aglet performs best; Voyager is better than Grasshopper.

B. Aglets:

Aglets are a Java mobile agent platform and library that eases the development of agent based applications. They can able to autonomously and spontaneously move from one host to another [23]. Originally developed at the IBM Tokyo Research Laboratory, the Aglets technology is now hosted at sourceforge.net as open source project, where it is distributed under the IBM Public License. Aglets are completely made in Java, granting a high portability of both the agents and the platform. This includes both a complete Java mobile agent platform, with a stand-alone server called Tahiti, and a library that allows developer to build mobile agents and to embed the Aglets technology in their applications.

Currently, stable release of Aglets is available in the 2.0 series, and 2.0.2 is the latest one. Aglet has been developed at the IBM Tokyo Research Laboratory (TRL) from Mitsuro Oshima and Danny Lange. The original name of the project was AWB that stands for Aglets WorkBench, changed then simply in Aglets. IBM was responsible for the most of the 1.x releases, while from the version 2.x Aglets is totally open source and is hosted at Sourceforge.net. The web page of the original project, still hosted at TRL, issues: Think of the Internet as a distributed, massively parallel supercomputer that connects information repositories, databases, intelligent agents, and mobile code. Imagine sending your own personalized agents to roam the Internet. They will monitor your favorite Web sites, get you the ticket you couldn't get at the box office, or help you to schedule meetings for your next overseas trip. Aglets are not the only one mobile agent development kit, but it is quite simple to learn and to use, and this probably helped its spread.

Aglets have been immediately involved in the realization of TabiCan, a kind of virtual agent-populated travel agency. Unfortunately, after a good start, IBM decided to give Aglets to the open source community, and this is when SourceForge appears. In the beginning, the SourceForge releases have been only bug-fix ones, but then something changed and the library version evolved to 2.x series. The 2.x thread has new improvements in the security management, and is more compatible with the Java 2 security mechanism than the 1.x releases. Furthermore, it includes a log4j based logging system and a few bug-fixes of the older version. After a couple of releases in the 2.x branch, the development stopped again. Now, starting from the 2.0.2 release, the development is going to restart, so stay tuned for newer versions.

C. Information Gathering:

The past several years have witnessed the rapid development of the World Wide Web (WWW). WWW has been a vast repository of information. Today, these Webs' size, dynamic and distribution nature put a significant amount of time and effort to locate, retrieve and integrate the desired information. Currently, Web crawlers, which traverse numerous webs by following hyperlinks and storing downloaded pages in a large database that is later indexed for efficient execution of user queries, are mainly used tools for information gathering. But there exist two limitations that can weaken crawlers' efficiency. One is even the largest and the most powerful search engines, such as Google, cover only limited parts of the webs. Another is many of the data that crawlers gathered is several months out of date.

With the popularity of MA technology, the information gathering system in combination with MA and web crawlers is more powerful and more efficient. Utilizing mobile agent technology, the information gathering system's capacity is improved. The proliferation of the diverse information in the internet makes the information gathering via a single mobile agent difficult. In addition, the limitations of the traditionally sequential task restrain the single agent to be applied to large, complicate applications. As a result, the applicable scope of a single mobile agent is limited. The advent of the collaborating multi-agents infrastructure not only alleviates this difficulty of gathering information throughout networks but also makes the ubiquitous internet computing possible.The in-depth example of the use of agents for an important class of problems i.e.` information gathering [24] is provides by He et al. They constructed a model of Web information gathering based on Mobile Agent technology, and the design of the model and the working process is introduced. A model of cooperating working for information gathering and a kind of arithmetic to implement this model are given. In addition, the upgrade of information gathered is implemented by making use of Mobile Agent technology. Aneiba and Rees addressed mobile agent as tools for mobile computing and these have been used in applications ranging from network management to automatic software distribution, as well as information management [25]. Mobile agent architecture is developed by Jonathan and DeRoure for distributed information management [26]. A brief overview and an elaborate case study for mobile agents [27] and their use for information retrieval are presented by Glitho et al.

The model of information gathering system is composed of many mobile agents and an agent server. The agent server mainly focuses on implementing the control of information gathering, management of the whole system, and the cooperative working of those mobile agents. The agent server's functions could be depicted as producing and initiating all mobile agents ,managing the execution of all mobile agents, guaranteeing the security of the mobile agents and receiving and processing the information sent by mobile agents, managing the saved Web information database. The functions of mobile agent in this system includes moving intelligently in specific network domain, crawling on the Webs, gathering the useful information, monitoring the change of Webs, recording the working status itself. The structure the Information Gathering Model and working process is detailed in [24].

In view of the popularity of MA technology for information gathering, the java based mobile agent systems such as Aglets is studied in the present study, for their strategic application in the area of integration and gathering of the information. The InfoGatherAgent java program is designed and implemented for gathering the information from the different ports of Tahiti to reduce the network load.

II. MATERIALS AND METHOD

Aglet is a very popular mobile system. It is designed especially for creating mobile applications and has a very complete and complex API for mobile agent. In the present study, Aglet SDK is used as the mobile agent platform for the development of secure data transfer. The Aglet Software Development Kit (ASDK) is an implementation of the Aglet API. It includes Aglet API packages, documentation, sample agents, and the Tahiti Aglet server. This Aglet Workbench works on JDK1.1 or higher versions. It is qualified to run on Win95/XP/NT and SPARC/Solaris 2.5 [28].

A. Running the Aglet Server:

For launching the Aglets first of all user need to start the aglets server, which can be started using the script file ‘c:\cd aglets2.0.2’. The login screen can be open by command C:\Aglets2.0.2>Agletsd.

B. Running Tahiti:

The aglet server will involve an aglet viewer, named Tahiti, for managing aglets. The user has to run the Aglets application through server called Tahiti. So, start it up with the agletsd command and create one of the provided agents (or aglets) to see whether everything works. Observe the MSDOS window. It should not display any error message (some non-error output is fine). The default port number as defined in aglets is port-4434. If you want to run an additional Tahiti server on the same machine, start it up with another port address, e.g., 2000.

C. Design of MA Program:

A simple agent consists of basically the main class and two methods on Creation () and run () [7]. First start by importing the aglet package, which contains all the definitions of the Aglet API. Next define the MyFirstAglet class, which inherits from the Aglet class:

```
Import com.ibm.aglet;
Public class MyFirstAglet extends Agent
        {
           // aglet's method here…….
        }
```

For example, if user wants your aglet to perform some specific initialization when it is created, user can override it’s on Creation method:

```
Public void on Creation (Object init)
        {
         //Do some initialization here…….
        }
```

When an aglet has been created or when it arrives in a new context, it is given its own thread of execution through a system invocation of its run method. The run method is called every time the aglet arrives at or is activated in a new context. So the run method becomes the main entry point for the aglet’s thread of execution.

```
Public void run ()
         {
           //Do something else here…..
          }
```

III. RESULTS AND DISCUSSION

The design of mobile agent and its implementation for the information gathering required to create two Tahiti servers for listening on different ports as described earlier. Two Tahiti servers are created using the script file ‘c:\cd aglets2.0.2’. The default port 4434 is created by Tahiti and for the creation of port 2000, it can be given as ‘c:\cd aglets2.0.2>’. Now, write as ‘c:\aglets2.0.2> agletsd-port 2000’. Similar procedure can be repeat for the creation of port 3000 as c:\aglets2.0.2>agletsd-port 3000’. After issuing the commands on command prompt, the following screen is displayed as shown in figure 1.

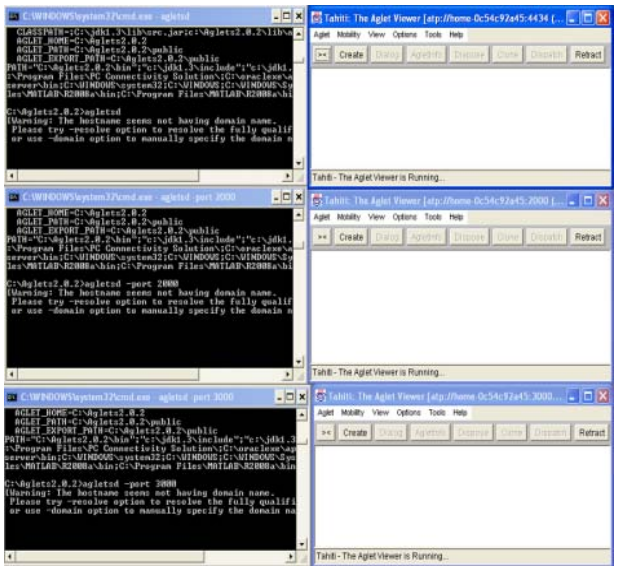

Figure 1. Tahiti Windows for Ports-4434, 2000, and 3000

Since, the main objective of the present study is to gather the information like user name, from different port address such as port 2000, port 3000, etc., the InfoGatherAgent.java program is created under jdk. The purpose of this program is to gather information from remote machines, which are part of a distributed system connected through LAN. InfoGatherAgent creates a Command Window GUI to obtain itinerary. Once URLs are given, visits each URL collects the required information and returns back to that URL where it has to display the result. Using Command Window displays the gathered information. The created 'InfoGatherAgent.java' program is stored in public subdirectory aglet 2.0.2 as shown in the figure 2.

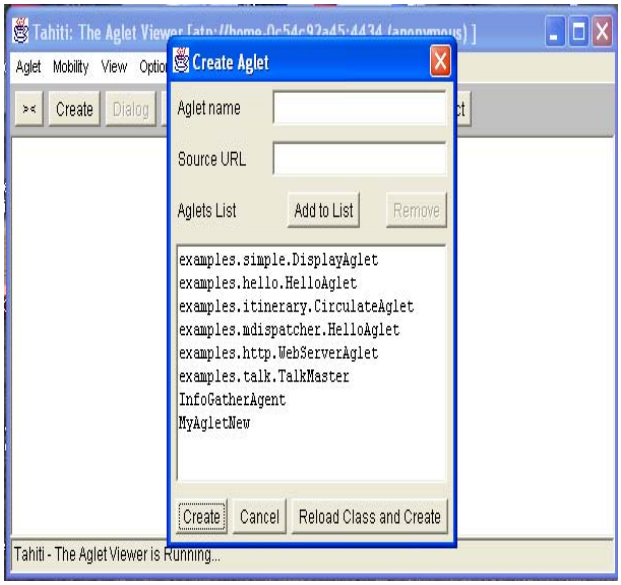

Figure 2. Agent Creation Dialog Window

Now, to gather the information from different ports of Tahiti server to default port address 4434, click on the create option of aglet menu. It displayed the list of programs from which the IngoGatherAgent program is selected. Click on create option at the bottom of the display box as shown in figure 3.

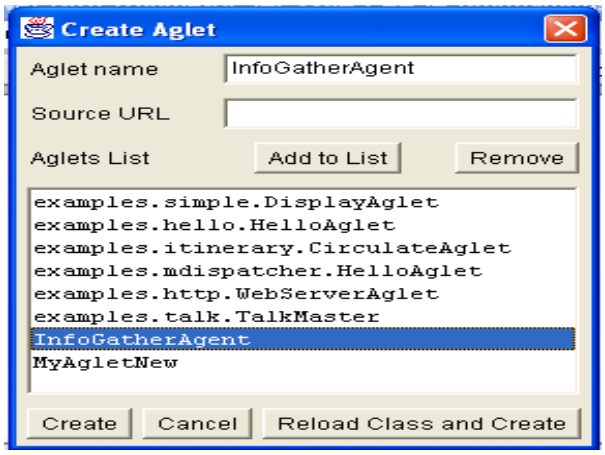

Figure 3. Agent list

After selecting the program "InfoGatherAgent" from create list, the following screen is displayed as shown in figure 4.

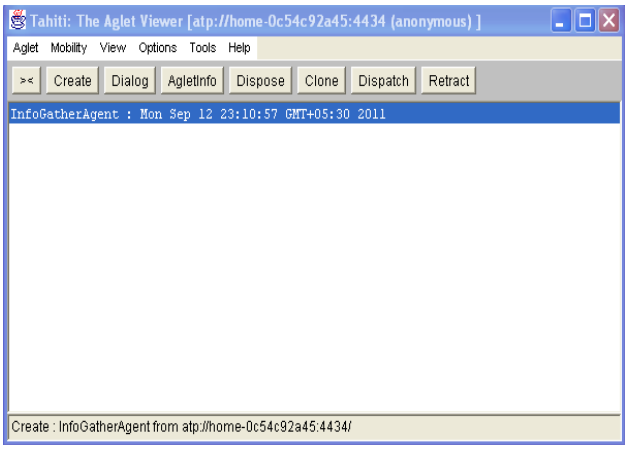

Figure 4. Tahiti selecting InfoGatherAgent

Now, click on Dialog menu to select port of the address from which we want to gather the information and to display the gathered information. This displayed the AddressBook for the selection of various ports addresses as shown in figure 5.

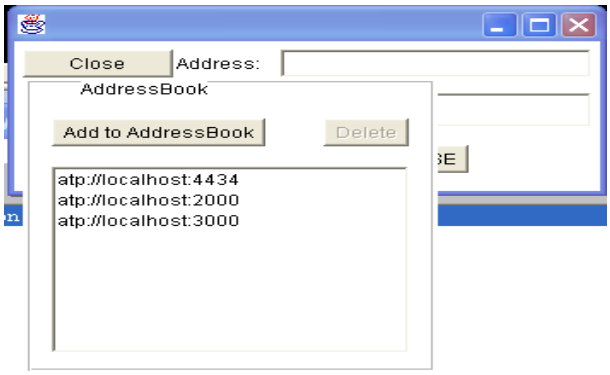

Figure 5. AddressBook Dialog Window

Next select the port -2000 from AddressBook, for getLocalInfo, and port-3000 for geLocalInfo, and port-4434 for printResult i.e. gather the information from port -2000 and port-3000 and print the gathered result on port-4434 as shown in figure 6 (a and b).

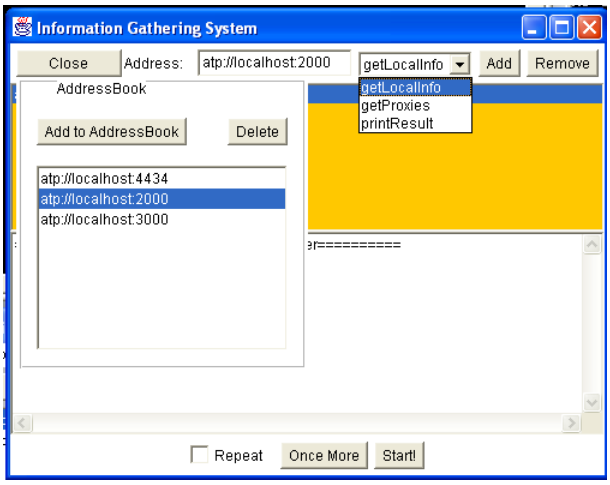

Figure 6 (a). Portaddress References Dialog Window

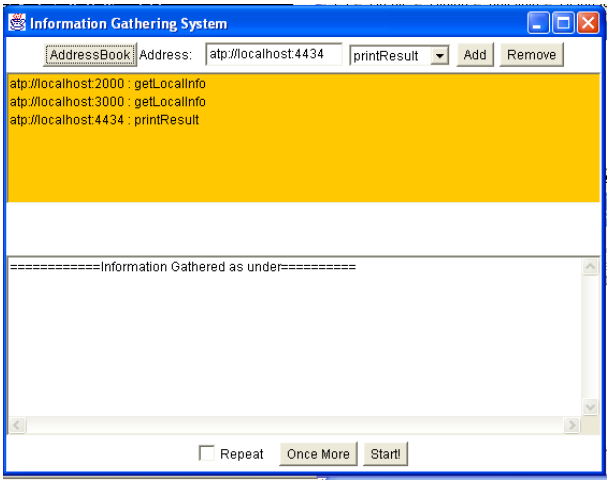

Figure 6 (b). Portaddress References Dialog Window (After Selection)

After completing the port selection criteria, click on the Start! button at the bottom of the portaddress reference dialog window. The InfoGatherAgent crawled to every port and gathered the information or data as per the InforGatherAgent.java program as shown in the figure 7 given below. Thus, the designed configuration of mobile agent as "InfoGatherAgent" gathered the information from different ports, which helps to reduce the network load and overcome the network latency.

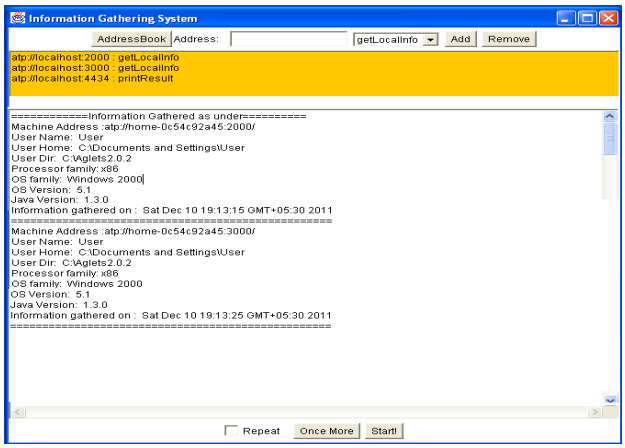

Figure 7. Portaddress References Dialog Window showing Informaation Gathered from Different Port Addresses.

IV. CONCLUSIONS

InfoGatherAgents program was designed, implemented and tested for information gathering from different ports. The designed InfoGatherAgents crawls to different ports e.g. Port 2000, Port 3000 for collecting the information about system status such as user name, user home, user directory, processor family, OS family, OS version, java version, etc. In the same fashion, the information from different ports can be gathered, when very large volumes of data are stored at remote hosts, that data should be processed in its locality rather than transferred over the network. Overall, the motto for InfoGatherAgent-based data processing is simple such as to move the computation to the data rather than the data to the computation. Thus, it can be useful for reducing the flow of raw data in the network.

V. REFERENCES

[1]. J. Gosling, B. Joy, B. and G. Steele. Java Language Specification. Addison-Wesley, Reading, MA, 1996.

[2]. S. Adnan, J. Datuin, P. Yalamanchili. Survey of Mobile Agent Systems. URL: http://www.cs.ucsd.edu/classes/sp00/cse221/reports/dat-yal-adn.pdf

[3]. N. Suri, M. Carvalho, R. Bradshaw, and J. M. Bradshaw. Small mobile agentplatforms.URL:http://autonomousagents.org/ubiquitousagents/papers/papers/32.pdf.

[4]. Zhaohui Hu. Mobile Agent Systems: An Overview. Research on Applications of Computer Technology. No. 10, 2000.

[5]. J. Cheng and V. Wei. Defenses Against the Truncation of Computation Results of Free-roaming Agents. Proceedings of 4th International Conference on Information and Communications Security, Lecture Notes in Computer Science, 2513: 1-12, 2002.

[6]. J. Zhou, J. Onieva and J. Lopez. Analysis of a Free Roaming Agent Result-Truncation Defense Scheme. Proceedings of 2004 IEEE Conference on Electronic Commerce, San Diego, USA, IEEE Computer Society Press, pages 221-226, 2004.

[7]. J. Zhou, J. Onieva and J. Lopez. Protecting Free Roaming Agents against Result-Truncation Attack. Proceedings of 60th IEEE Vehicular Technology Conference, Los Angles, USA, pages 3271-3274, 2004.

[8]. D. Lange, and M. Oshima. Seven good reasons for mobile agents. Communications of the ACM, ISSN: 0001-0782, 42(3), March 1999.

[9]. Maria Fasli. Agent Technology for e-Commerce - Chapter 11: Mobile Agents. URL:http://cswww.essex.ac.uk/staff/mfasli/ATe-Commerce.htm

[10]. J. White. Telescript Technology: Mobile Agents, General Magic White Paper. URL: http://www.genmagic.com/

[11]. J. Ousterhout, "Agent TCL", 1997, URL:http://agent.cs.dartmouth.edu/ general/ agenttcl.html

[12]. H. Peine, T. Stolpmann. The Architecture of the Ara platform for mobile Agents. Proc. of the First International

Workshop on Mobile Agents MA'97, Berlin, Springer Verlag, April 7-8, 1997.

[13]. D. Johansen,V. Renesse, and F. Schneider. Operating system support for mobile agents. Proc. of the 5th. IEEE HOTOS Workshop, Orcas Island, USA, 4-5 May, 1995.

[14]. Aglets platform. 2004. URL: http://www.research.ibm.com/trl/ aglets/spec10.htm

[15]. Concordia. Concordia: An Infrastructure for Collaborating Mobile Agents. Proc. of Workshop On mobile agents MA'97, Berlin, LNCS 1219, Springer Verlag, April 7-8th, 1997.

[16]. Java tutorial available at www.java.sun.com/tutorials

[17]. M. Straßer, J. Baumann, and F. Hohl. Mole - A java Based Mobile Agent System. Proc. ECOOP'1996 workshop on Mobile Object Systems.

[18]. N. Minar, M. Gray, O. Roup, R. Krikorian, and P. Maes. Hive: Distributed Agents for Networking Things. Proceedings of ASA/MA'1999.

[19]. Sun Microsystems Inc. Enterprise Javabeans Specification. Version 1.1, 1999, available at http://java.sun.com/products/ ejb/docs.html

[20]. K. Arnold, A. Wollrath, B. O'Sullivan, R. Sheifler and J. Waldo. The Jini Specification. Addison-Wesley, 1999.

[21]. D. B. Lange, and M. Oshima. Mobile Agents with Java: The Aglet API. World Wide Web Journal, 1998.

[22]. Zhao Qunhua, Wang Hua and Zhang Yi. Comparison Study of Three Mobile Agent Systems Aglets, Grasshopper and Voyager. Department of Computer Science and Engineering University of Connecticut.

[23]. Aglet, 2004, http://www.aglets. sourceforge.net/

[24]. Yongchun He, Cong Wang and Jian Qiu. An Information Gathering Model Based on Mobile Agents. IEEE, pages 225-228, 2005.

[25]. Adel Aneiba and S. J. Rees. Mobile Agents Technology and Mobility. Staffordshire University, PO BOX 334, Beaconside, Stafford ST18 ODG, UK.

[26]. D. Jonathan, and D. DeRoure. A Mobile Agent Architecture for Distributed Information Management. Proceedings of the International Workshop on the Virtual Multicomputerm, March 1997.

[27]. R. H. Glitho, E. Olougouna, and S. Pierre. Mobile Agents and Their Use for Information Retrieval: A Brief Overview and an Elaborate Case Study. Ericsson Res., Montreal, Que IEEE, 16(1):34-41, (ISSN: 0890-8044, INSPEC Accession Number: 7165883), Jan/Feb 2002.

[28]. Aglets framework available at www.trl.ibm.co.jp/aglets.