IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 14, Issue 6 (Sep. - Oct. 2013), PP 01-06
www.iosrjournals.org

A framework for an Operating System-based Mobile Agent Interoperability

Oyatokun Bosede O[1], Osofisan A.O[2] and Aderounmu, G.A[3]
[1]*Department of Mathematical Sciences, Redeemer's University, Mowe, Ogun State, Nigeria*
[2]*Department of Computer Science, University of Ibadan, Ibadan, Oyo State, Nigeria*
[3]*Department of Computer Science and Engineering, Obafemi Awolowo University, Ile-ife, Osun State, Nigeria*

Abstract: *Mobile agent technology has grown in acceptance over the years for distributed applications, but it is yet to be adopted as ubiquitous solution technique. This is due to its complexity and lack of interoperability. Mobile agent executes on mobile agent platform, these platforms from different vendors are design, and language specific, and are thus non interoperable. In other words mobile agent built on one platform cannot interact with or execute on any other platform. There is a need to provide a common base on which agents from different vendors can interact and interoperate. This work presents a framework for mobile agent interoperability by providing an Embedded Mobile Agent (EMA) system into the Windows Operating System kernel so that it can run as a service; this was done to eliminate the overheads associated with the agent platforms and enhance mobile agents' interoperability. The targeted OS were Windows XP, Windows Vista and Windows7.*

Keywords: *embedded mobile agent, interoperability, mobile agent platform, operating system service.*

I. Introduction

Mobile agent paradigm has been recognized as a viable tool and a promising approach for building distributed applications [1,2] and a lot of research has been done, nevertheless, it is still a promising area of research, because, a lot of its many potentials are yet to be exploited. Mobile agents are autonomous software capable of performing computational tasks on behalf of another software or human user [1, 2, 3]. Mobile agent is defined as a computer entity capable of reasoning, use the network infrastructure to run in another remote site, search and gather the results, cooperate with other sites and return to its home site after completing the assigned tasks [4]. Mobile agents paradigm provides infrastructure for executing automous agents and also migrate them between computers connected by a network. Mobile agent paradigm is made up of two prominent components, the mobile agent itself and the mobile agent middleware system called the mobile agent platform. The mobile agent platform provides the physical infrastructure in which agents are deployed [5]. Mobile agent platforms is the execution environment for agents, and provides functionalities that support migration of agents, communication between agents, various programming languages and various forms of security [6].

II. Mobile Agent Systems

Over the years, several mobile agent platforms have been developed to support mobile agent applications [6,7,8]; these platforms operate independent of one another which hinders the interoperability of mobile agents. The platforms are different in design, goals, language and vendor, thus they are not interoperable. In other words an agent designed on one platform cannot execute on another platform, neither can it interact with an agent from other platforms. Most agent platforms either offer enormous flexibility at the cost of usability or extended built-in functionality at the expense of interoperability [9].
We examine some of the existing agent platforms briefly in this section

- JADE: written in java and uses Message Transport protocols (HTTP, IIOP, HTTPS) for communication and migration.[5,10].
- TACOMA: Tromso And Cornell Moving Agent (developed by University of Tromso, Norway & Cornell University, NY), written in TCL (Tool Command Language) but can carry script in other languages [6].
- Aglet: is a combination of Agent and Applet, written in Java programming language and uses HTTP for communication [11].
- Agent TCL (D'Agent): created at Dartmouth College, the platform is written in C and the agent in TCL, it uses proprietary protocol over TCP/IP and PGP [12].
- Telescript/odyssey: Telescript is an object oriented scripting language for implementing mobile agents, it implements strong migration (**agent go** to **place**) [13]. Telescript was later implemented in java and was called odyssey.

- Voyager: is java-based and agent-enhanced Object Request Broker (ORB). Voyager communicates through RMI (Remote Method Invocation) using proxies, uses TCP/IP for migration; it is commercial product with free license allowing non-commercial use of its core technology [6].
- Grasshopper: complies with MASIF and FIPA standards, it is implemented in java and supports TCP/IP, RMI/JRMP and CORBA/IIOP [14].
- Mole: developed in java, uses RMI for communication[6]

The development of these agent platforms is motivated by different goals which include support for specific agent models, programming environments, mobility and security [7].

Agents need to communicate with one another in the process of working together to achieve a common goal, agent paradigm of software development believes that communities of agents are much more powerful than any single agent, which necessitates interoperation of agent systems. Interoperability in mobile agent community focuses on the execution environment and standardization of certain aspects and features of agents while in the non-mobile agent contex the focus is on communication, i.e. effective exchange of information and knowledge content of agents. Interoperability has been defined by [15] as follows: two mobile agent systems are interoperable if a mobile agent of one system can migrate to the second system, the agent can interact and communicate with other agents (local or even remote agents), the agent can leave this system, and it can resume its execution on the next interoperable system.

III. Mobile Agent Interoperability

A lot of research work is presently going on in the area of mobile agents interoperability [15,16,17] several solutions have been proposed but they lack the necessary flexibibility to provide adequate degree of interoperability among the available MASs. Interoperability is paramount to the global acceptance of mobile agent system (MAS) in heterogeneous and open distributed environments where agents must interact with other agents to fulfil their tasks and visit different agent platforms to access remote resources [17].When mobile agents migrate to a new host, the platform on the host provides execution environment, the mobile agent might execute code, make remote procedure calls to access resources on the host, collect data or initiate another migration process. Problems arise from the fact that not all platforms for mobile agents are the same and thus, cannot provide necessary services for non-compliant mobile agents[6]. Interoperability is directed at making an agent system accept and support the running of agents from another agent system and vendor, support the transfer of agent to other agent systems and find other agents and agent systems. To this effect, efforts have been made to standardize certain aspect of mobile agent paradigm such as agent management, agent transfer, agent and agent system name.

3.1 Mobile Agent Paradigm Standards

The Foundation for Intelligent Physical Agent (FIPA) and Mobile Agent System Interoperability Facility (MASIF) have made efforts to define sets of standards for mobile agents and agents' platform [18Zeghache]. FIPA addresses the interoperability among agents, attempt to standardize certain aspects of mobile agent and defines features of agents such as communication, agent management and the agent abstract architecture [5]. MASIF addresses the interoperability between agents' platforms, attempts to standardize some aspects of the execution environment to provide for mobile agents to interoperate and it focuses on agent management, agent transfer and name for agents and agent platform [5,19]. MASIF as defined by the Object Management Group (OMG) consists of a collection of definitions and interfaces that provides interoperability among mobile agent systems; it provides two interfaces; the MAFAgentSystem for agent transfer and MAFFinder for naming and locating [19]. These efforts are yet to be effective at providing the necessary interoperability among agents and agent systems.

3.2 Previous works

Interoperability Application Programming Interface (IAPI) that supports registration, lookup, messaging, launching and migration of agent across different patforms was proposed in [16]. The system provides three layers to the GMAS layer, the Foreign2GMAS translator, GMAS2Native translator and common communication and discovery service. The system only enabled agent migration among diverse agent platforms but the agents may fail to execute due to difference in the level of the java API. The additional software layers constitute a significant overhead, at the same time, the performance of the system was also slow, the additional layers on the platforms being the major factor.

A java-based framework for interoperability among java-based mobile agent systems was proposed by [20]. The framework permits interoperability of execution, migration and interaction of java-based mobile agent systems. The framework consists of three software layers, the Interoperable Mobile Agent Layer (IMAL), the Adaptation Layer (AL) and the Platform-dependent Mobile Agent Layer (PMAL) which constitute a

considerable overhead. Ata the same time, a Mobile Agent Bridge must be developed for each agent platform to be able to migrate, this constitutes an additional overhead on the system.

Secure and Open Mobile Agent (SOMA) [21] is another attempt at achieving interoperability; it was developed in compliance with both CORBA (Common Object Request Broker Architecture) and MASIF. SOMA uses a CORBABridge which consists of CORBA client/server which simplifies the design of SOMA entities as CORBA client /server and MASIFBridge which implements the MASIF functionality. The security and fault tolerance of the system is important for interoperability to be fully attained, SOMA achieves security but it is not fault tolerant. Moreover, the MASIFBridge introduced a considerable overhead and the model has a close connection with CORBA which limits its application.

Agent operating system (AOS) designed by [7], provides common primitives required by most agent platforms so they can interoperate, AOS was a portable and language-neutral middleware that resides between the agent platform and the operating system. AOS facilitates interoperability between agent platforms and between different implementations of AOS itself. The AOS provides a common interface for different agent platforms to execute in order to achieve interoperability, in other words it provides a meeting point for the agent platforms and does not attempt to eliminate agent platforms. The AOS contribute another overhead to the system.

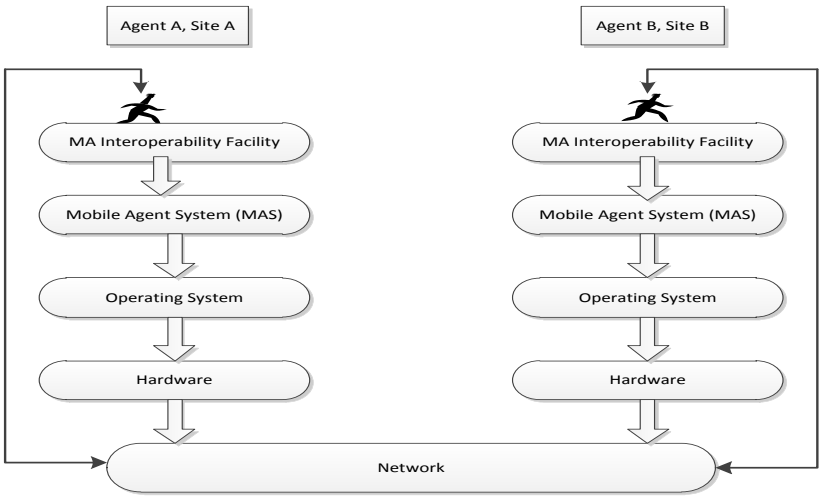

Figure 1: the conceptual model of existing platform-based mobile agent system

The existing interoperability models provide a facility over the mobile agent platform for interoperability as shown in figure 1, which constitutes a significant overhead to the overall system. The shortcomings of the above interoperability models led to our attempt to find a common platform on which agents from different platforms and vendors with different design and architecture can communicate, execute and interact effectively and efficiently without fear of risk or vulnerability to failure and other attacks. Several mobile agent platforms have been developed by different groups, although these agent platforms differ in their goals, designs, motivations and implementations, they all provide common functionalities that support: agents' migration, agents' communication, various programming and interpreted language and various forms of security [6]. This work is an attempt to provide such stage on which agents from different vendors can interoperate without necessarily going through the agent platform.

IV. Architecture of the Proposed Framework

The proposed system consists of a lightweight static agent embedded into the kernel of the windows operating system in the form of a service as a Terminate and Stay Resident (TSR) program. The static agent is installed as part of the executive services in the kernel mode of the Windows operating system. Windows (XP and higher versions) operating system provides a mechanism to make certain user programs run in its kernel mode giving an impression of programming the operating system. In the actual sense of it, the services of the operating system are being extended.

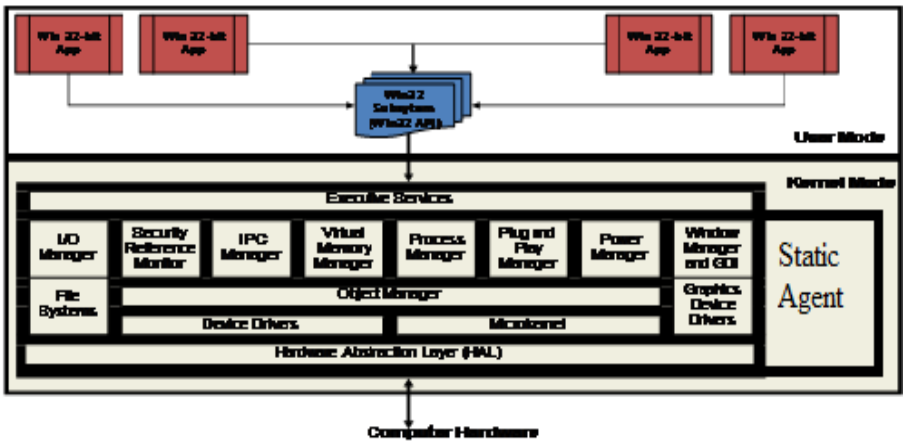

Fig.2: Structure of the proposed embedded agent in Windows XP (adapted from [22])

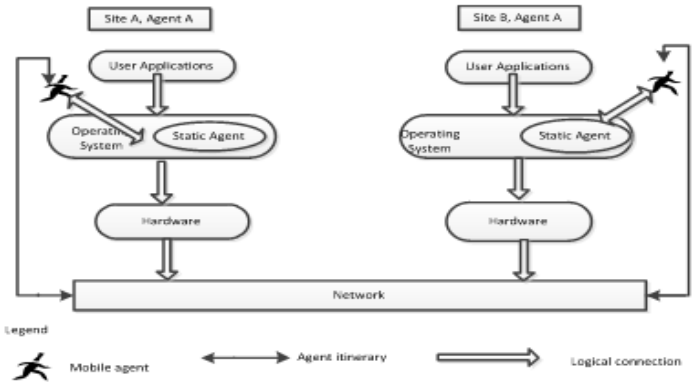

Figure 3: The Conceptual diagram of the proposed Embedded Mobile Agent

Mobile agent from remote host interacts with the static agent in the kernel mode of the visited host operating system, giving an impression of directly interacting with the operating system.

4.1 The proposed system concepts

The static agent executes on the host where it begins execution performs a number of functions related to information storage and retrieval.

- It is responsible for listening to the port for incoming agent.
- It negotiates passage to the destination host and ensures that the mobile search agent is successfully transferred. If the mobile search agent is rejected, it restarts the agent to allow it choose another destination.
- It validates and authenticates the incoming agent
- It launches received mobile agents and provides runtime execution for the mobile agent according to the level of trust given to the agent. The runtime execution environment will depend on the access level granted to the mobile agent and the functions it wishes to perform.
- It provides a registration to register mobile agents and hosts on the network with the available resources on them.

4.2 Operation

The static agent on the remote host authenticates and receives in comming mobile agent, searches its local files for the relevant information, and then downloads the information and forwards it to the mobile agent to add it as part of its bag. The mobile agent moves to the next host in its itinerary. On reaching a new host in its itineray, the mobile repeats the same process and moves to the next host until the last node in its itinerary. It then returns home with the results in its bag and forward the result to the static agent who displays the result to the user.

- Incoming Mobile Agent seeks permission to perform its tasks. The static agent receives and authenticates the incoming agent.
- The static agent after receiving the requests interprets the requests and searches the local database for available relevant documents, it queries the database using keywords

- The static agent adds the downloaded documents to the mobile agent as part of its bag.
- The mobile agent saves its current state, signs off the visited node, exit and continue in its itinerary, and if it's the last node in its list, returns to the origin, delivers the result and disposes itself.

V. Conclusion

This work proposes a framework for mobile agent interoperability by providing an Embedded Mobile Agent (EMA) system into the Windows Operating System kernel so that it can run as an operating system service. This mode of design and deployment eliminates the use of agent platform which for a long time has been the limiting factor for mobile agents' interoperability. The work is built upon the earlier work of the authors in [23] where mobile agent was deployed as an operating system service. This work is similar but different from other proposals made in the past for mobile agents' interoperability. The main focus of [18] is the standardization issues for agent interoperability; it integrates two standards (MASIF and FIPA-ACL) to propose an architectural model for mobile agent system interoperability. The focus of [24] is on agent architecture, it separates all platform specific code from platform independent main procedure of an agent, so that the agent can migrate to an incompatible platform. The agents however, cannot use all the features of the underlying platform. In [16] an Interoperability Application Programming Interface (IAPI) built on top of the agent platform serving as a translator between agents and the platforms was proposed. Secure and Open Mobile Agent, SOMA [20] focuses on standardization for achieving agents' interoperability, SOMA was developed to comply with both CORBA and MASIF. The Agent Operating System (AOS) by [7] focuses on interaction between agent platforms and provides a set of primitives that are common to agent platform. The AOS was designed as a portable middleware layer between the mobile agent platform and the operating system and facilitate interoperability between agent platforms. Our approach to interoperability however, focuses on the mode of deployment of mobile agent. A light weight static agent is embedded into the kernel of the operating system as an operating system service and provides runtime execution for mobile agents thereby, eliminating the need for an agent platform. The framework has been implemented in java programming language and testes on the Windows XP, Windows vista and Windows 7. Work is ongoing in implementing the framework on other operating systems, specifically Unix and any of its flavour, as well as provision of adequate security for the system. In addition, the implementation of mobile agents from other vendors and platforms is a continuation of this work.

References

[1] Lange D. B. 1998. Mobile objects and mobile agents: the future of distributed computing? Proceedings of the European Conference on Object-Oriented Programming (ECOOP'98), 1998.

[2] Aderounmu, G. A. 2001 Development of an intelligent mobile agent for computer network performance management. Unpublished PhD thesis, Department of Computer Science and Engineering, Obafemi Awolowo University, Ile-Ife, Nigeria.

[3] Biermann E. 2004. A Framework for the Protection of Mobile Agents Against Malicious Hosts. Unpublished Ph.D thesis, University of South Africa, South Africa.

[4] Outtagarts Abdelkader (2009). "Mobile Agent-Based Applications: A Survey." *International Journal of Computer Science and Network Security, 331 - 339.*

[5] Bellifemine, F.L, Greenwood D and Caire G (2007). Developing Multi-agent systems with JADE. John Wiley & Sons Ltd, England.

[6] Syed A., John D. and Pavana Y. (2000). A survey of mobile agent systems. Student Report, Department of Computer Science and Engineering, University of California San Diego. (Date of last access: 03 July, 2013 from cseweb.ucsd.edu/classes/sp00/cse221/reports/dat-yal-and.pdf).

[7] van 't Noordende, G.J., Overeinder, B.J., Timmer, R.J., Brazier, F.M.T. and Tanenbaum, A.S. (2009). Constructing secure mobile agent systems using the agent operating system', *International Journal of Intelligent Information and Database Systems*, Vol. 3, No. 4, pp.363–381.

[8] Gabriel S. and Claudiu I.P. 2010. A Proposal for an Enhanced Mobile Agent Architecture (EMA). *Annals of the University of Craiova, Mathematics and Computer Science Series, 37(1): 71 – 79.*

[9] Tudor M., Bogdan D., Mihaela D., Ioan S. 2004. A Framework of Reusable Structures for Mobile agent Development. *Proceedings of the 8th IEEE international Conference on Intelligent Engineering Systems (INES '04), Cluj-Napoca, 2004.*

[10] Giovanni C. 2009. JADE Programming for Beginners. TILAB, S.P.A.

[11] Venners B. 1997. Under the hood: The architecture of aglets. Java-World, January 1997.

[12] Robert S. G. (1997). Agent Tcl: a flexible and secure mobile-agent system. Unpublished PhD thesis in Computer Science at Dartmouth College, Hanover, New Hampshire.

[13] General Magic. 1995. Telescript language Reference. October,1995

[14] Gupta R. and Kansal G. 2011. A survey on comparative study of mobile agent platforms. *International journal of engineering, science and technology* (IJEST), 3(3): 1943 - 1948.

[15] Pinsdorf U and Roth V. (2000). Mobile Agent Interoperability Patterns and Practice. Available at http://jade.tilab.com/papers/EXP/pinsdorf.pdf.

[16] Grimstrup A., Gray R., Kotz D., Breedy M., Carvalho M., Cowin T., Chacon D., Barton J., Garrett C. and Hofmann M. 2002. Toward interoperability of mobile agent system. *Proceedings of the sixth IEEE international Conference on Mobile Agent, Bercelona, Spain, 106-120.*

[17] Labrou Y, Fini T and Peng Y. 1999. The Interoperability problem: Bringing together mobile agents and agent communication languages. Proceedings of the Hawaii International Conference on System Sciences, copyright 1999 IEEE.

[18] Zeghache L., Badache N., and Elmaouhab A. (2004). An Architectural Model for a Mobile Agents System Interoperability.

[19] Milojicic D., Breugst M., Busse I., Campbell J., Covaci S., Friedman B., Kosaka K., Lange D., Ono K., Oshima M., Tham C.. Virdhagriswaran S. and While J. MASIF: The OMG Mobile agent System Interoperability Facility, personal technologies, 2(3), 117-129, December, 1999.
[20] Bellavista P., Corradi A. and Stefanelli C. (2001). Mobile agent middleware for mobile computing. IEEE Computer Society, Washington DC, USA, 73-8.
[21] Fortino G. and Russo W. 2003. High-level interoperability between java-based mobile agent systems. A report of the project ' Giovane Ricercatore 2003', University of Calabria.
[22] WIN133. 2009. The big picture….which makes more sense now. Retrieved on 20 *September*, 2012.
[23] Oyatokun B. O, Osofisan A. O. and Aderounmu G. A. 2013. An Operating System-based model for mobile agent deployment. International Journal of Computer Science and Information Security (IJCSIS), 11(8), 92-96. ISSN 1947-5500.
[24] Pauli M., and Kimmo R. (2000). Agent Migration between incompatible agent Platforms. p*roceedings of the 20th IEEE International Conference on Distributed Computing Systems (ICDCS'00,), : 4*.