Mobile Agent Infrastructures: a Solution for Management or a problem to Manage?

Paulo Simões, Luís Moura e Silva, Fernando Boavida

CISUC – Dep. Engenharia Informática,

Universidade de Coimbra, Pólo II, P-3030 Coimbra, Portugal

Abstract

Mobile Agent Technology (MAT) is a fresh paradigm for distributed programming, with potential for application in a broad range of fields [1]. A Mobile Agent (MA) corresponds to a small program that is able to migrate to some remote machine, where it is able to execute some function or collect some relevant data and then migrate to other machines in order to accomplish another task. The basic idea of this paradigm is to distribute the processing throughout the network: that is, send the code to the data instead of bringing the data to the code. MA systems differ from other mobile code and agent-based technologies because increased code and state mobility allow for even more flexible and dynamic solutions.

Distributed network management is one of the most frequently mentioned application fields for MAT [2-4]. One of the main reasons for this enthusiasm over MAT is the perception that the classical management architectures do not cope well with the increasing complexity of nowadays systems. In order to be managed, they require the level of flexibility and dynamic adaptation that MAT will presumably provide.

However, one must not forget that the deployment of MAT will also increase even more the complexity of the system to be managed. This is a case where the management solution, if not carefully handled, ends up as another management problem. This issue is often overlooked in the design of MA systems, with dramatic consequences on the costs of installation and administration of the distributed infrastructure that is required by mobile agents.

Although every mobile agent system provides some sort of mechanisms to control the agent's lifecycle and to monitor the state of the supporting platform, we feel there is a general lack of manageability in current systems. This weakness is twofold: offered functionality is not just unsatisfactory but also unreachable from external applications.

In the last couple of years we have developed the JAMES platform for mobile agents [5], which was later used by our industrial partners to produce and deploy several MA-based applications for telecommunications and network management. These circumstances provided us with a good perspective on infrastructure manageability and lead to the development of several services for better administration of the JAMES platform.

This paper presents those services and discusses their impact on the reduction of the associated management costs.

The rest of the paper is organized as follows: Section I discusses the manageability problems of current MA implementations. Section II provides an overview of the JAMES management mechanisms, and Sections III, IV and V describe individually each of those mechanisms. Section VI discusses related work, and Section VII concludes the paper.

I. TYPICAL MANAGEMENT SCENARIO

Figure 1 depicts the most frequent scenario in the management of mobile agent systems, with two distinct management points.

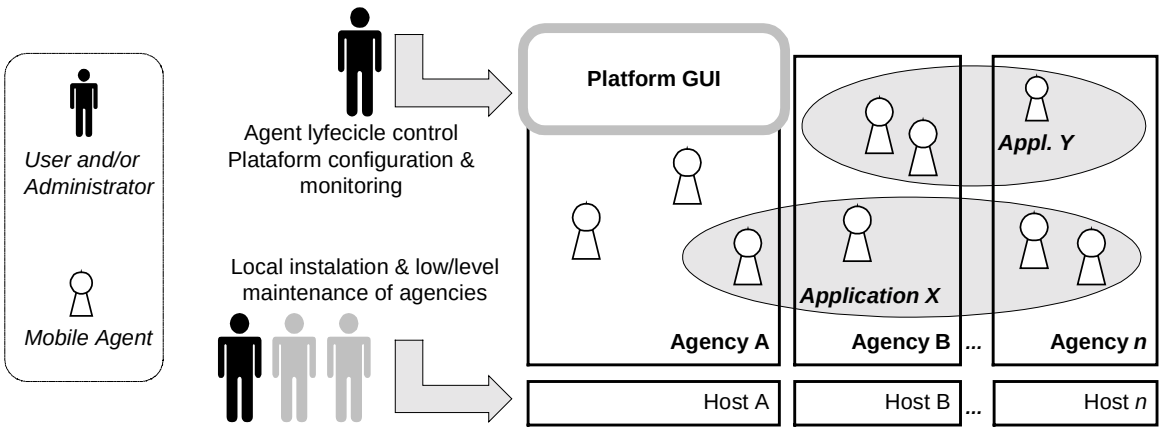

Fig. 1 Typical Management on Current Mobile Agent Systems

The first point is related with the local installation of the platform in every node of the infrastructure. In order to support mobile agents, a computer must have some kind of supporting environment, usually provided by a java program that we will generically designate as the *agency*. This agency needs to be installed, upgraded, controlled and – whenever failure occurs – recovered. This kind of low-level maintenance is not automated by the majority of mobile agent systems, and therefore requires costly local human intervention. This is unacceptable whenever there are a large number of *agencies* to maintain or the network nodes are geographically dispersed.

The second management point in typical mobile agent systems is a platform-attached graphical interface that is used to launch and to control mobile agents, to receive their results, and to configure and monitor the whole MA infrastructure. The interface is designed both for the manager and the final user, which are often expected to be the same person.

There are several problems with this user interface. First, it mixes infrastructure management with MA usage, which are two separate issues. Secondly, it assumes that managers and users are always humans, excluding the possibility of having outside applications managing and using MA-based services. Thirdly, they do not comply with open management standards, leaving the MA infrastructure as a strange body outside the scope of integrated managed systems. Last but not least, they often provide just a limited set of the desired functionality and flexibility.

II. THE JAMES MANAGEMENT ARCHITECTURE

In order to tackle with these problems, three different meta-management services were introduced in the JAMES platform (Figure 2).

The first is a simple low-level service that runs at the host's operating system level and controls the execution of the local agency, providing software upgrade, monitoring and restart services from remote locations.

The second service is the *Remote API*, which consists on a single unified interface for interaction between the mobile agent system and external applications that use or manage the JAMES platform. The *Remote API* replaces the traditional platform GUI with a more automated interface, suitable for applications rather than humans.

In order to comply with integrated management systems, an SNMP [6] management service was also developed. This service corresponds to an SNMP extensible agent and a specifically defined MIB – the JAMES-MIB – that represents the meta-management functionality of the JAMES platform according to the SNMP information model.

Each of these services will be individually described in the next Sections.

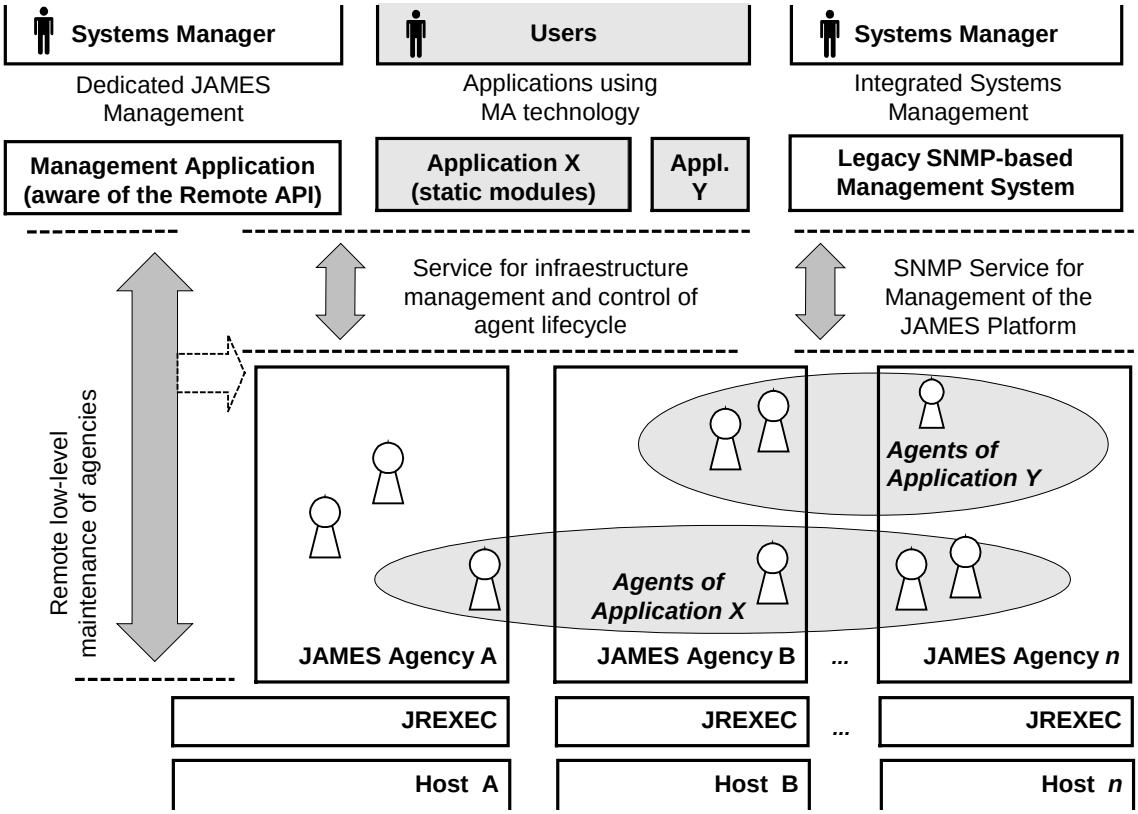

Fig. 2 JAMES Services for Meta-Management

III. REMOTE ADMINISTRATION OF THE AGENCIES

One of the most attractive advantages of designing applications using MAT is precisely the ability to dynamically control and upgrade remote applications. Since those applications consist of mobile agents, this is simply a matter of replacing the older agents by corrected or upgraded agents. However, there is still one remaining application that requires local maintenance: the agency itself. Since this agency is a relatively complex computer program, the need to manually recover from abnormal faults or to perform a software upgrade is still more frequent than desirable.

To minimize this problem, the JAMES agency was split in two different modules: a very small and very stable module that requires almost no maintenance at all, and a larger application (the *real* agency) whose execution is controlled by the smaller module.

This module, designate as *JREXEC* is a daemon permanently running at the host's operation system level. This daemon performs several functions:

- start the execution of the agency;
- check the agency status;
- refresh the local agency;
- stop the agency;
- and upgrade the agency, receiving its new code from a remote location.

As illustrated in Figure 3, the JREXEC service is available both to external management applications and to the other nodes of the JAMES infrastructure (in the context of internal management).

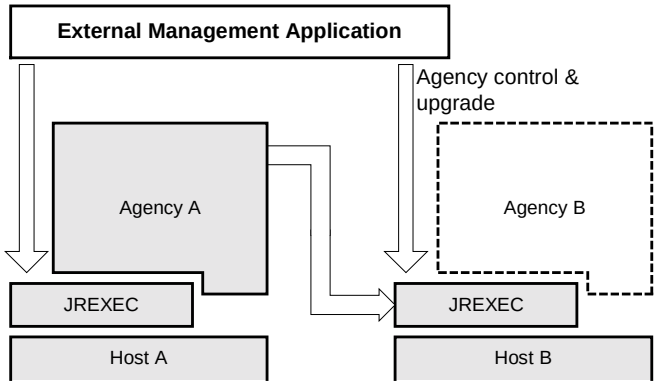

Fig. 3 The *JREXEC* Service

With few exceptions, current MA platforms do not provide similar services and require either local installation and maintenance (with the already mentioned increase in running costs) or the usage of general remote desktop applications like Microsoft's Systems Management Server or Intel's Landesk. These tools, however, are too much system dependent and do not integrate well with internal platform management mechanisms. It should be noted that JREXEC itself is not absolutely portable: in order to comply with the hosting operating system (e.g. installation as Windows NT Service or Unix Daemon) minor adjustments have to be made from system to system. However, the JREXEC interface remains constant from system to system.

IV. THE REMOTE API

As already mentioned, the interfaces provided with the majority of MA implementations are designed with humans in mind – it is assumed that humans are directly using or managing the MA infrastructure. This results in several when one tries to integrate MAT into larger application frameworks or into integrated management systems.

In the JAMES Platform a different approach was used, with the provision of a single unified programming interface for interaction between the mobile agent system and external applications. This interface covers all aspects, from the basic configuration and management of the platform to the creation and control of mobile agents. This solution provides a superior degree of automation and integration for MAT, and also simplifies the whole designed of the platform, since all user interfaces were removed (and replaced by external applications that provide the same kind of functionality for human users).

The Remote API is built using Java RMI, and consists of two modules (Figure 4). The first is part of the agency and acts as a mediator between the communication layer and the platform's core functionality. The second module is integrated within the external application, providing a high-level remote interface to access the platform. The communication layer between these two modules remains hidden from the application developer. The current implementation of the application-related module is based on Java RMI, but other technologies, like CORBA, could easily be included.

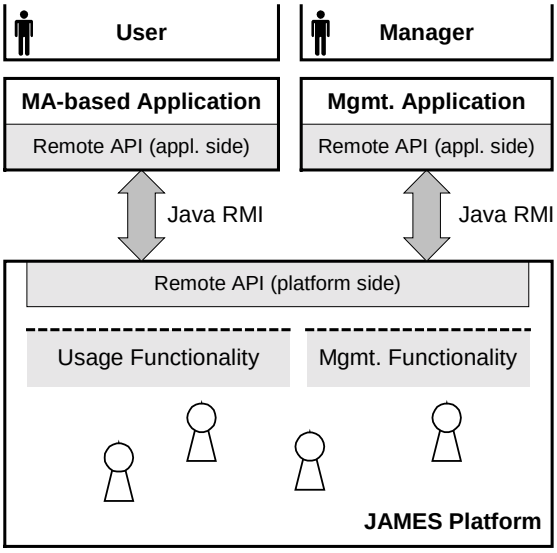

Fig. 4 *Remote API*

The programming interface reflects the administrative model of the JAMES Platform, supporting flexible coexistence between different users and different applications with different security permissions.

Other platforms share some of the concepts behind the *Remote API* [7-8], but their implementation does not provide the same level of functional decoupling.

The *Remote API* approach is also similar to the MASIF specification [9], which defines a set of CORBA interfaces for interoperability between mobile agent systems. Using MASIF interfaces external applications are able to launch, locate and control the lifecycle of agents or even to add new agencies to the infrastructure. The *Remote API* could have been designed as an extension of the MASIF specification, since its functionality is basically a superset of MASIF. However, despite the positive reception of this standardization initiative, only two products currently comply with MASIF [10-11]. For this reason, the extra effort necessary to convert the *Remote API* model to the more generic MASIF model was considered to be not worthwhile.

V. THE JAMES-MIB FOR META-MANAGEMENT

The Remote API should be considered as the primary interface point to manage the JAMES infrastructure. However, it requires specifically designed management applications. In order to achieve integration with generic management systems another kind of interface – based on open standards like SNMP [6] – is required.

The JAMES Platform already entails a complete framework for integration with SNMP [12], and therefore adding an SNMP service for the management of JAMES infrastructure was a simple task.

First, a new MIB, designated as JAMES-MIB, was defined in order to represent the management functionality to be made available. This MIB converts a subset of the *Remote API* management functionality to the SNMP information model, including tables to represent the status of agencies, mobile agents, applications and users (Figure 5). Complete translation of the *Remote API* was not considered given the SNMP limitations in the representation of complex management functionality.

Since the JAMES platform already includes an optional extensible SNMP service [13] compliant with the AgentX standard [14], implementing the JAMES-MIB was a straightforward task. An SNMP/AgentX management subagent was created to virtually implement this MIB. It receives SNMP/AgentX requests; translates them into JAMES internal management actions and then answers back to the SNMP managers. Figure 6 represents the JAMES-MIB subagent and the SNMP/AgentX master service.

In order to demonstrate the JAMES-MIB functionality a specific SNMP-based application was built, but the service is most useful in the context of integration with legacy management platforms like HP Openview [15].

As far as we know there is only another experience addressing SNMP-based administration of mobile agent systems. In this experience [16] the functionality of MASIF was translated into an SNMP-MIB implemented by an external adaptor. With this adaptor MASIF-compliant platforms can be accessed from SNMP-based applications.

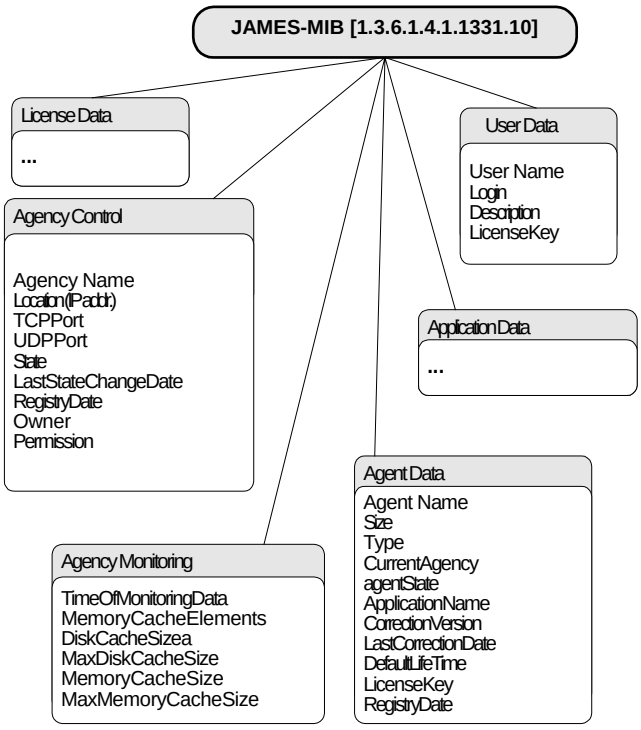

Fig. 5 JAMES-MIB Main Tables

VI. CONCLUSIONS

Manageability is a key factor in the success of mobile agent systems. Without proper support for infrastructure management the costs of deploying and exploring MA-based applications easily override the advantages that would justify the introduction of novel technology. For this reason one would expect that mobile agent implementations would be designed with this requirement in mind. However, this is seldom the case.

In this paper we presented a set of mechanisms that were added to the JAMES platform to enhance its manageability. The first is a very simple yet effective mechanism that deals with the low-level administration of remote agencies, drastically reducing the need for local intervention. The second corresponds to a unified interface for external applications. This interface replaces the usual human-oriented GUIs with a more application-oriented service, providing better integration between mobile agents and other distributed computing technologies. The third consists of an SNMP service available to legacy management applications. With this service the JAMES infrastructure becomes a generic SNMP-compliant another managed object.

VII. ACKNOWLEDGEMENTS

The JAMES project was partially supported by Agência de Inovação (ADI) and was accepted in the European Eureka Program (Σ!1921).

Special thanks to Eduardo Lourenço, Pedro Pereira and Rodrigo Reis, for the development of the SNMP services, to Luis Santos, for the development of JREXEC module, to

Paulo Martins, Victor Batista and Guilherme Soares, for the development of the Remote API, and to the rest of the Project Team.

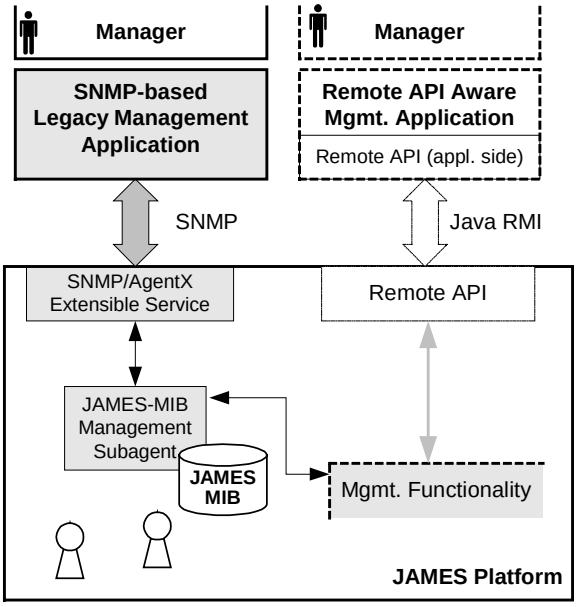

Fig. 6 Architecture of the JAMES-MIB Service

VIII. REFERENCES

[1] V. Pham, A. Karmouch, "Mobile Software Agents: An Overview", IEEE Communications Magazine, July 1998

[2] A, Bieszczad, B. Pagurek, T. White, "Mobile Agents for Network Management", IEEE Communications Surveys, 4th quarter 1998, Vol. 1, No. 1, pp. 2-8

[3] D. Gavalas, D. Greenwood, M. Ghanbari, M. O'Mahony, "Advanced network monitoring applications based on mobile/intelligent agent technology", Journal of Computer Communications, Vol. 23, Issue 8, pp. 720-730, Elsevier, Abril de 2000

[4] M. Zapf, K. Herrmann und K. Geihs, "Decentralized SNMP Management with Mobile Agents", Proceedings of the IM'99, Boston/USA, 1999

[5] L. Silva, P. Simões, G. Soares, P. Martins, V. Batista, C. Renato, L. Almeida, N. Stohr, "JAMES: A Platform of Mobile Agents for the Management of Telecommunication Networks", Proceedings of IATA'99, Stockholm, 1999

[6] M. Rose, "The Simple Book - An Introduction to Management of TCP/IP-based Internets, 2nd Edition", Prentice-Hall International Inc., 1994

[7] Objectspace Voyager, `http://www.objectspace.com/voyager/`

[8] Jumping Beans, `http://www.JumpingBeans.com/`

[9] "Mobile Agent System Interoperability Facilities Specification", OMG TC Document orbos/97-10-05, 1998

[10] IKV++ Grasshopper, `http://www.ikv.de/`

[11] A. Puliafito, O. Tomarchio, L. Vita, "MAP: Design and Implementation of a Mobile Agents Platform", Journal of System Architecture, 46(2):145-162, 2000

[12] P. Simões, L. Silva, F. Boavida, "Integrating SNMP into a Mobile Agent Infrastructure", Proceedings of DSOM'99 – Tenth IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, Zurique, 1999

[13] P. Simões, E. Lourenço, P. Pereira, L. Silva, F. Boavida, "J.AgentX: a Tool for Dynamic Deployment of Open Management Services", in Proceedings of SoftCom'2000, Split, October, 2000

[14] M. Daniele, B Wijnen, D. Francisco, "Agent Extensibility (AgentX) Protocol Version 1", RFC 2257, 1998

[15] Hewlett-Packard Openview Homepage, `http://www.openview.hp.com`

[16] R. Lopes, J. Oliveira, "Descrição e Implementação de Uma MIB para Sistemas MASIF", Actas da 3ª Conferência sobre Redes de *Computadores*: Tecnologias e Aplicações (CRC'2000), Viseu, 2000