ENABLING MOBILE AGENT TECHNOLOGY FOR LEGACY NETWORK MANAGEMENT FRAMEWORKS

Paulo Simões, Rodrigo Reis, Luís M. Silva, Fernando Boavida

University of Coimbra, CISUC — Dep. Eng. Informática
Pólo II, Pinhal de Marrocos
P-3030 Coimbra, Portugal
`psimoes@dei.uc.pt`

Abstract. *Network management is often considered as one of the application areas with greatest potential for Mobile Agent (MA) technology. However, legacy applications and architectures impose considerable inertia to the deployment of new solutions for network management. For this reason, successful MA infrastructures will need flexible and effortless integration with legacy management frameworks, like the widespread SNMP architecture.*

This paper presents a set of solutions for the integration of SNMP into MA platforms, to support access to SNMP management resources and to introduce MA-based management services that can be reachable from legacy SNMP-capable applications. Such solutions were successfully applied to the `JAMES` *platform, a joint project from University of Coimbra, Siemens Portugal and Siemens AG.*

KEYWORDS: Network Management, Mobile Agents

1. INTRODUCTION

Network Management (NM) applications are usually based in one of two classic protocols: SNMP [1], widely deployed in IP networks, and CMIP [2], for telecommunication networks. These protocols are based on static, centralized and non-scalable client/server architectures, where some central entity processes large amounts of raw data gathered from each Network Element (NE).

The need for more scalable and flexible NM applications is leading to an intensive quest for higher decentralization [3], with approaches like Management by Delegation, CORBA, Web-based management, Intelligent Agents, Active Networks and, more recently, Mobile Agent Technology (MAT) [4].

A Mobile Agent (MA) can be described as a small software program that is able to migrate between hosting computers during its execution whilst maintaining across the network. With this, the task processing can be dynamically distributed through the network and placed closer to the management data. When compared to classic client-server solutions, MAT reduces network traffic and increases scalability, flexibility and robustness. MAT has been applied to several areas, like mobile computing, e-commerce, Internet services, information search, and network management [5].

There is now a considerable number of commercial MA platforms, like IBM's Aglets [6], Mitsubishi's Concordia [7], General Magic's Odyssey [8], ObjectSpace's Voyager [9], AdAstra's JumpingBeans [10] and Grasshopper [11], from IKV++. However, despite their interesting features, these platforms are too much general-purpose and do not provide any special support for network management.

For this reason, in the `JAMES` Project [12], we are developing from scratch a new MA infrastructure that is being tuned and customized for the applications we have in mind in the area of telecommunications and network management. Although the discussion of the platform issues is beyond the scope of this paper, we can point out some particular aspects like efficient code migration; fault-tolerance and robustness; flexible code distribution and easy upgrading; mechanisms for resource control; disconnected operation; portability; and interoperability with existing management technologies, like SNMP [13].

This interoperability is crucial because legacy management technologies are the link to provide the access to the management services available in the network. MAT provides powerful programming metaphors that allow more efficient solutions for distributed network management. However, one still has to rely on the old management protocols in order to access the management resources in heterogeneous environments, whenever direct Java interfaces are not available or managed resources can not host mobile agents.

Additionally, MA-based solutions often need to coexist with and integrate into legacy management systems. It is much more attractive and cost-effective to develop and deploy management services using MAT if those services are usable by installed legacy applications. This kind of interoperability, that can be achieved by equipping MA-based services with SNMP or CORBA interfaces, paves the way for incremental and integrated introduction of MAT to solve specific problems for legacy management frameworks. Without it, MAs are limited to specialized and poorly integrated management applications.

It is possible to use currently available general-purpose SNMP tools to achieve some degree of interoperability between MAT and SNMP. However, there are several reasons that recommend explicit support from the MA infrastructure, like code mobility constraints, security constraints, limitations imposed on the programming model of mobile agents, and resource usage control.

This paper proposes a framework to integrate SNMP into MA platforms. We designed this framework specifically for the `JAMES` platform. However, since the features it requires from the platform are relatively current, it can be easily be ported to most of the other MA platforms.

The rest of this paper is organized as follows: Section 2 presents the `JAMES` project, and Section 3 presents the integration of SNMP into the `JAMES` platform. Section 4 discusses related work, and Section 5 concludes the paper.

2. THE JAMES PROJECT

The JAMES project explores the use of MAT in NM applications and consists of two complementary lines of work. First, the development of a MA platform specifically tuned according to the requirements those applications impose (coordinated by University of Coimbra). Second, the development of software products that exploit the technological and economical advantages of the platform (coordinated by Siemens Portugal and Siemens AG). In the next Subsection, we will present a brief description of the MA platform. An overview of the first software products developed over the MA platform can be found in [14].

2.1 General Architecture of the JAMES Platform

The JAMES Platform provides the running environment for MA. In its present version, there is a distinction between the software environment that runs in the manager host (the JAMES Manager) and the software that executes in the other network nodes (designated as JAMES Agency). Figure 1 shows a global overview of the system.

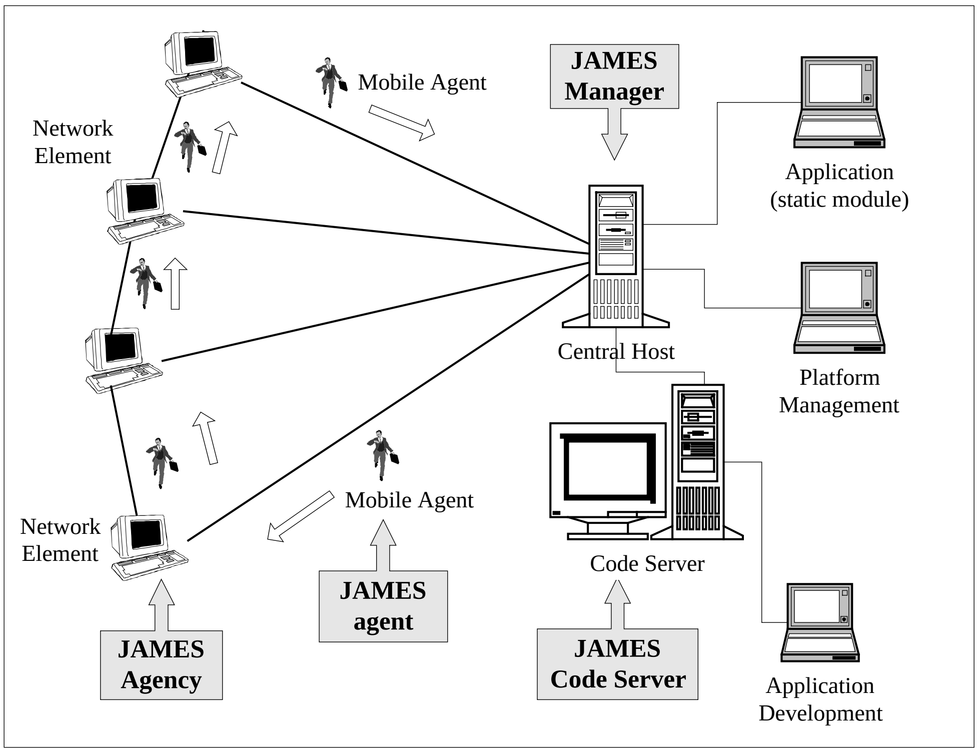

Figure 1 — An Overview of the JAMES Platform

The JAMES Manager is responsible for the maintenance of the whole MA infrastructure. Each NE runs a Java Virtual Machine and executes a JAMES Agency that enables the execution of the mobile agents. The JAMES agents will migrate through these machines of the network to access some data, execute some tasks and to exchange information with other agents or other applications. There is a mechanism of authentication in the JAMES Agencies to control the execution of agents and to avoid the intrusion of non-authorized agents.

The applications consist of one or several MAs, and may additionally include classic “static” programs coordinating the MA activities or providing a Graphical User Interface to the end user. These static external modules communicate with the agents using a Remote API, which also allows the remote management of the platform.

MAs are developed in Java and use the JAMES API for the control of mobility. After development, MAs must be registered and stored in the JAMES Code Server. This server keeps a relation of all authorized MAs, as well as other security-related information. For scalability reasons, future versions of the platform will support multiple Code Servers and JAMES Managers.

The explanation of the inner details of the JAMES platform is outside the scope of this paper. However, in the following list we summarize the key features of our system:

- Portability of the applications, through the use of the Java language;
- High-performance in mobility through the use of caching and prefetching techniques;
- Security mechanisms for code authentication;
- Resource control service to manage the use of underlying resources (CPU, memory, disk and operating system resources);
- System monitoring and profiling of agent activity;
- Fault-Tolerance through the use of checkpointing and dynamic reconfiguration of itinerary;
- Easy-to-use programming interface;
- Scalable execution of mobile agents, through the use of decentralized protocols;
- Remote “on-the-fly” software upgrading;
- Interface with CORBA services;
- Support for Java-based technologies, like JavaSpaces [15];
- Distributed management and easy configuration of the network;
- Inter-agent communication (through Javaspaces);
- Multi-paradigms for agent execution (simple agent, migratory agents and Master/Worker model);
- And integration of Java-based SNMP services into the platform.

2.2 The JAMES interoperability with SNMP

The JAMES platform provides support for SNMP in a bi-directional way (see Figure 2), leading to three distinct services: interaction with local and remote SNMP agents; access to MA-based services from legacy applications; and a service allowing legacy applications to manage the JAMES platform itself.

Additionally, several key design goals were defined:

- Transparency: for obvious reasons, no interventions should be required on the installed SNMP devices or applications;
- Minimal impact on the JAMES infrastructure: SNMP competes with the *Remote API* as an interface for upper applications, and with several other tools available to access management resources. Therefore, solutions where the platform portability, complexity or functionality are affected by SNMP support are not acceptable;
- Full support for agent mobility: meaning that SNMP support should not restrict agent migration.

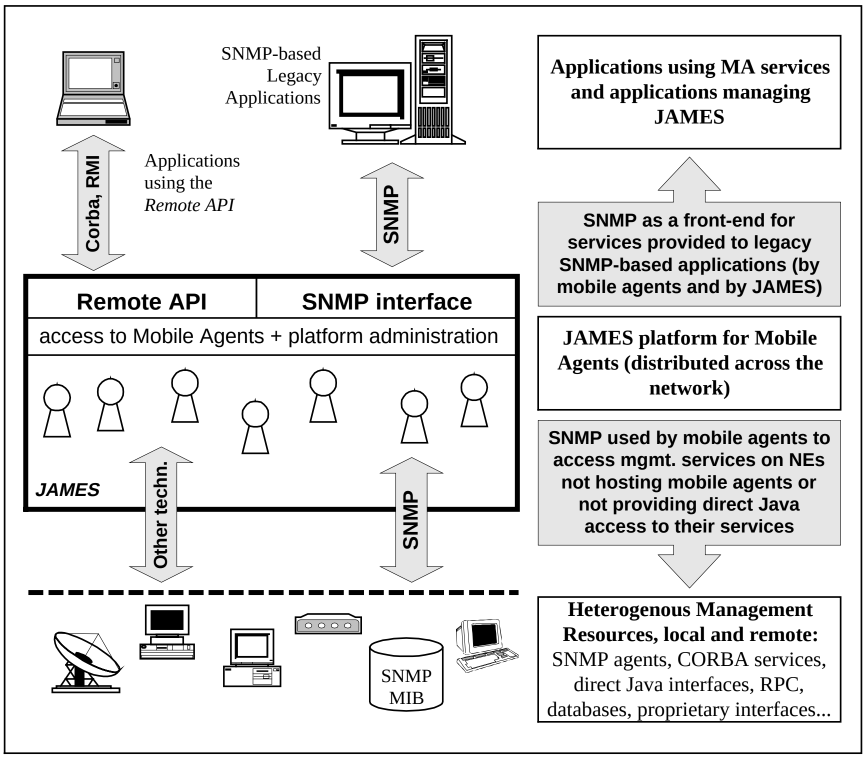
SNMP-based Legacy Applications
Applications using the *Remote API*
Corba, RMI
SNMP
Applications using MA services and applications managing JAMES
SNMP as a front-end for services provided to legacy SNMP-based applications (by mobile agents and by JAMES)
Remote API
SNMP interface
access to Mobile Agents + platform administration
JAMES platform for Mobile Agents (distributed across the network)
JAMES
Other techn.
SNMP
SNMP used by mobile agents to access mgmt. services on NEs not hosting mobile agents or not providing direct Java access to their services
SNMP MIB
Heterogenous Management Resources, local and remote: SNMP agents, CORBA services, direct Java interfaces, RPC, databases, proprietary interfaces...

Figure 2 —Bi-directional SNMP Interoperability

3. THE JAMES SNMP EXTENSIONS

The design of the SNMP Services of JAMES tries to accomplish with the defined design goals (cf. Subsection 2.2) defining a modular framework where each SNMP service can be installed and removed “on-the-fly”, according to the circumstances (see Figure 3).

Most services consist themselves of mobile agents (the *Service Agents*) providing services to common mobile agents through inter-agent communication. These Service Agents, that can be located (or implicitly installed) using a general directory service, are a lightweight solution that elegantly replaces more sophisticated component-based technologies in the provision of several kinds of services.

3.1 SNMP-Manager Service (SMS)

This service allows MAs to interact with SNMP Agents, using an SNMP manager-API to query SNMP agents, and a Trap Listener that receives SNMP-Traps and redirects them to the interested MAs. The basic functionality that is provided is not too different from similar services found in classical management applications, with concepts like *sessions* or *contexts*, *request* operations, asynchronous API, event handlers and registration of Trap interest. However, there are two key differences: the service location within the platform, based on the Service Agent concept, and support for agent mobility — management operations using the SMS do not affect the mobility of the MA, since all messages and events are transparently forwarded to the MA current location.

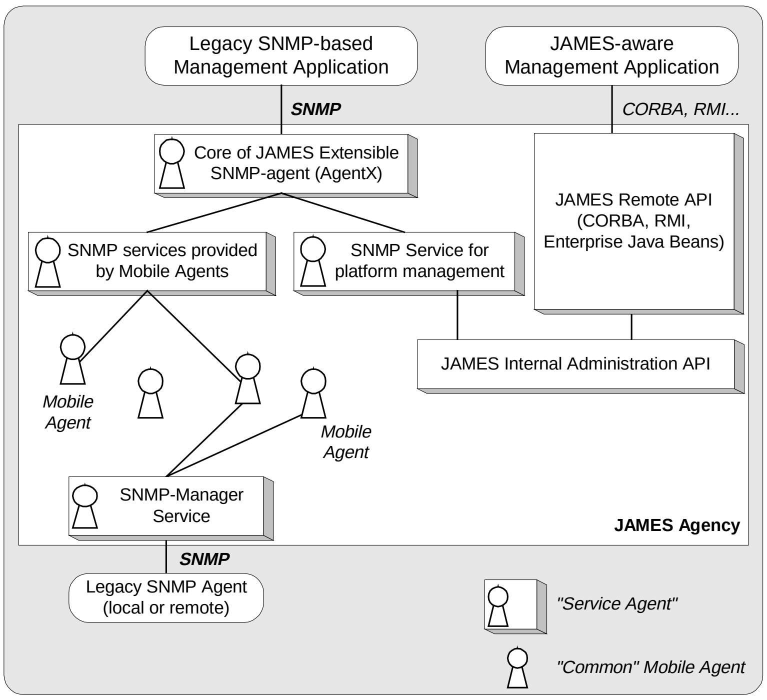
Legacy SNMP-based Management Application
JAMES-aware Management Application
SNMP
CORBA, RMI...
Core of JAMES Extensible SNMP-agent (AgentX)
JAMES Remote API (CORBA, RMI, Enterprise Java Beans)
SNMP services provided by Mobile Agents
SNMP Service for platform management
JAMES Internal Administration API
Mobile Agent
Mobile Agent
SNMP-Manager Service
JAMES Agency
SNMP
Legacy SNMP Agent (local or remote)
"Service Agent"
"Common" Mobile Agent

Figure 3 — High-Level Structure of JAMES SNMP Services

There are however many situations where such a complete support for agent mobility is not necessary. Indeed, when Trap reception is not an issue, the Service Agent based SMS might be replaced by a third-party "classic" SNMP stack integrated in the Agent's code, trading-off mobility support. This possible trade-off is based on the assumption that mobile agents can delay migration whenever completion of on-going SNMP transactions is crucial, since they implicitly control their migration.

In this alternative (that we will designate as "Fat Agent Model", opposed to the previous described "Service Agent Model") the SNMP manager-API is part of the MA code, increasing its size. Mobility is affected by this increase, leading to higher latency when migrating between agencies. Furthermore, MAs may loose incoming asynchronous events (like SNMP traps or responses) when they migrate to another agency. The programming model is also affected (agent migration has to become aware of SNMP transactions) but still allows for some mobility. Figure 4 presents the main differences between both alternatives.

The JAMES platform also includes an SMS based on a “Fat Agent” Model, to be used in situations where the more complete “Service Agent” model is not necessary or recommended.

A third alternative, the location of the SMS in the core of the JAMES agency (eventually designated as “Fat Agency Model”) was not considered because we wanted to keep minimal impact on the platform. However, with more sophisticated component-based technology this could become an interesting alternative.

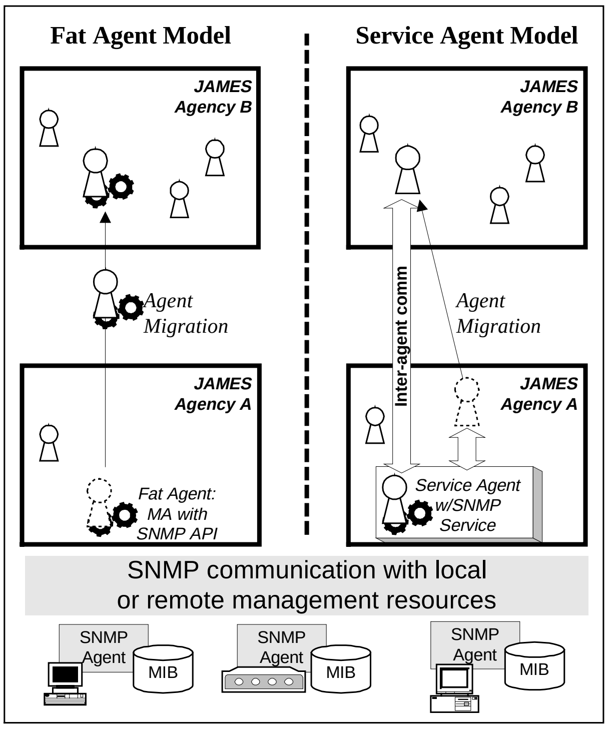

Figure 4 — Service Agent Model vs. Fat Agent Model

3.2 SNMP Services for Legacy Applications

The interface with legacy SNMP Managers is provided by three different services, also located on Service Agents (see Figure 5).

The Extensible SNMP-agent of JAMES consists of a core SNMP-agent with data being supplied by the two underlying services. The present interface used to register new variables or groups at the MIB-table, to issue Traps and to reply to SNMP requests is proprietary, but the standard AgentX [16] protocol is currently being adapted.

This SNMP-agent is independent of eventually installed native SNMP-agents. Since native agents are either monolithic or based on a wide diversity of agent-expansion mechanisms, like DPI [17] or AgentX, there was no truly portable and non-intrusive integration method. This option of separating the SNMP services provided by the JAMES mobile agents from the native SNMP-agents imposes some constraints. For instance, it is not possible to extend a MIB of the native

SNMP-agent. However, this is not problematic since the most usual use of mobile agents, in this context, will be the provision of new services and not a very localized expansion of existing SNMP Services.

The JAMES SNMP-agent allows SNMP communication between legacy applications and mobile agents, opening the way for easy installation of new management services (corresponding to one or several mobile agents) available to legacy SNMP-based network management systems. In such a scenario mobile agents can be used to pre-process data gathered from existing management services (thus offering higher level functionality), operate as SNMP proxies for NEs using proprietary management interfaces, and can also install, dynamically, new management services.

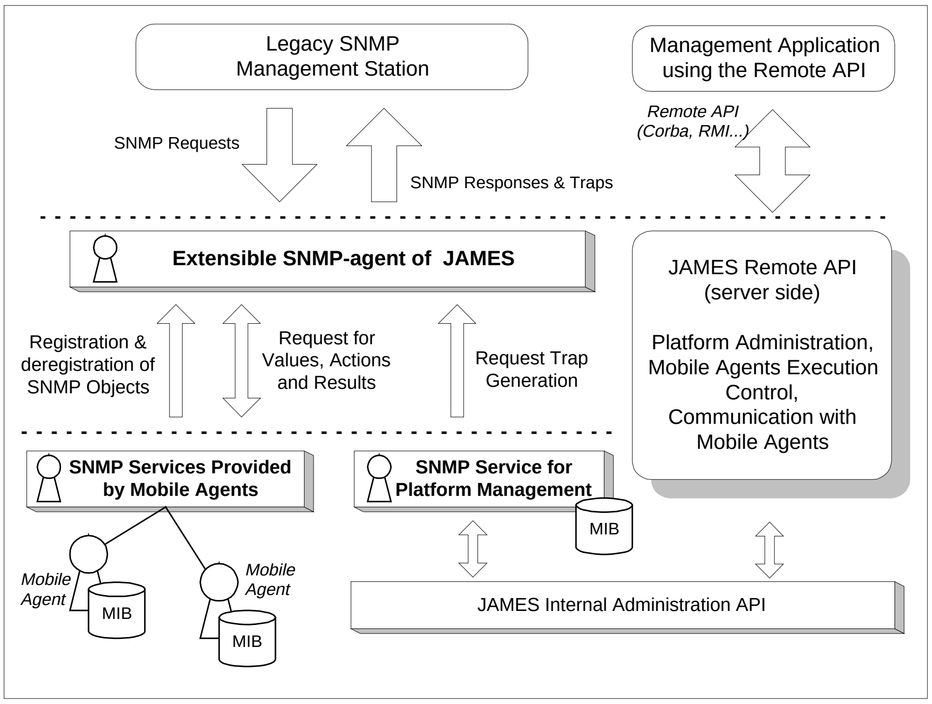
Legacy SNMP Management Station
Management Application using the Remote API
SNMP Requests
SNMP Responses & Traps
Remote API (Corba, RMI...)
Extensible SNMP-agent of JAMES
JAMES Remote API (server side)
Platform Administration, Mobile Agents Execution Control, Communication with Mobile Agents
Registration & deregistration of SNMP Objects
Request for Values, Actions and Results
Request Trap Generation
SNMP Services Provided by Mobile Agents
SNMP Service for Platform Management
MIB
Mobile Agent
MIB
Mobile Agent
MIB
JAMES Internal Administration API

Figure 5 — Services for Integration with Legacy Applications

Bellow the Extensible SNMP-agent, there is a mediator service where mobile agents interested on providing an SNMP interface must register their SNMP objects. Later on, SNMP requests from outside applications result in events passed to mobile agents. These events will then trigger predefined management actions resulting in SNMP responses. With the adaptation of AgentX, this mediator will no longer be necessary, and interest mobile agents will communicate directly with the extensible SNMP-agent.

The idea of using mobile, stationary or extensible agents for fast deployment of management services available to legacy applications is not new. The use of mobile agents and DPI to provide services to SNMP Managers was proposed by [18]. In the field of Management by Delegation a similar approach, based on extensible agents, was also considered [19], and the JDMK toolkit [20] offers a complete set of tools to create and remotely install new management services based on stationary services.

The JAMES SNMP agent also allows SNMP-based management of the JAMES platform itself. Although a richer interface is available to dedicated applications using the *Remote API*, a subset of the available management functions has been "translated" into an SNMP MIB that provides monitoring, fault-management and performance management. This is implemented using another "Service Agent" (the Agency SNMP Management Service) that mediates the communication between the SNMP-agent and the internal JAMES administration API. The intention is not to use SNMP to fully administer JAMES but to provide a basic set of SNMP-based management services.

For the moment, there are two separate security frameworks: one for the SNMP interface, with the basic mechanisms of SNMPv1 and SNMPv2c, and another for the *Remote API*, with finer granularity security policies (per-application and per-agent) and stronger authentication mechanisms. The current SNMP agent of the platform does not reflect this finer granularity by dynamically creating a matching SNMP "view" for each JAMES application that uses the *Remote API* interface. For this reason, we feel this area still requires further work.

3.3 Implementation Notes

The JAMES SNMP services were completely developed from scratch. However, there are no strong technical reasons excluding the less costly extensive adaptation of commercially available SNMP tools written in Java, like [21]. SNMPv1 and SNMPv2c are currently supported and, for the moment, implementation of SNMPv3 is not being considered.

The first version of these services used some basic inter-agent communication schemes, and is now being ported to a more flexible scheme based on Sun JavaSpaces. This work, as well as the already mentioned adaptation of AgentX, should be complete within a few weeks.

3.4 Portability to other MA Platforms

Extending this framework to other MA platforms should be possible, since most of them already include the main necessary features.

Porting the SNMP-Manager Service would simply require good asynchronous inter-agent communication mechanisms and a basic directory service to locate the corresponding Service Agents. For platforms already featuring full-fledged component-based design, it could be interesting to substitute these Service Agents by Service Components installable on the platform. The "Fat Agent Model" is even less demanding, since the service requires no support from the platform.

The service that allows creation of SNMP services using MAs is also simple to install using the same techniques of the SMS.

The service that allows to perform some platform management functions from legacy applications is the most difficult to port, for two major reasons: first, most MA platforms provide no mechanism like the *Remote API* and are rather based on thigh user interfaces. For this reason, most of them do not have a well-defined internal management API; secondly, even for the platforms that provide such an API, it would obviously be necessary to create a new MIB structure reflecting their own organization and security framework.

4. RELATED WORK

Unlike CORBA, supported by several platforms and with two on-going standardization processes (MASIF [22] and FIPA [23]), SNMP is not explicitly supported by any commercial implementation of MA. The few registered experiences belong to the academic field, with several projects [24-26] mentioning SNMP as an *ad-hoc* tool to access management services on network devices, without concern about specific integration issues. There are, however, two MA projects that, like `JAMES`, went further on the support (or usage) of SNMP.

The Astrolog project [27], that uses MAT to enhance the mobility of the network operator, included internal monolithic SNMP-agents to represent/access local information and to pass management data up to the remote controlling applications. These SNMP-agents, however, were developed with a single application in mind, lacking extensibility or flexibility.

The *Perpetuum Mobile Procura* Project [18,28] is the one that presents most similarities with the work made on our project. It also addresses bi-directional SNMP integration (managed resources and management applications), using DPI [17] to indirectly access local SNMP agents and to extend these agents. However, the use of an external DPI-capable SNMP agent affects the global portability of the system.

The *Java Dynamic Management Kit* (JDMK) [20] also deserves to be mentioned, since this commercial product from Sun Microsystems seems to be the most complete and powerful toolkit for the development of management services. It is based on a modular infrastructure that integrates several management protocols, including SNMP, with mechanisms for easy management service extensibility. JDMK shares several concepts with MAT, but it should be stressed out that JDMK agents are not MAs, even if it is possible to dynamically install them on the network devices using push/pull mechanisms.

5. CONCLUSIONS AND FUTURE WORK

In this paper we discussed the provision of explicit SNMP support for management applications based on mobile agents. This kind of support is important whenever SNMP is the only or the best available interface to access management information. Another reason to provide SNMP support is the possibility of using mobile agent technology to develop new management services to be used by legacy management applications. Furthermore, SNMP can also be used to manage the platform itself.

To the best of our knowledge our proposal presents the most comprehensive approach in the integration of SNMP with Mobile Agent Technology. We use mobile agents for accessing management services, we use SNMP to provide new services to legacy management applications, and we can even use SNMP for the management of the platform. None of the other platforms provides these facilities.

In addition to this extensive use of SNMP, the `JAMES` approach also provides enhanced support for MA mobility and, through the use of the "Service Agent" concept, presents minimal impact on complexity and performance of the platform.

We are currently conducting a comparison study between the Fat Agent Model and the Service Agent Model, considering mobility support, performance and scalability. We are also preparing an extensive performance study about the introduction of MA-based SNMP services for legacy management applications.

Another area where additional work is necessary is security, in order to cope with the already mentioned discrepancies between the security models of the *Remote API* and the SNMP interface for platform management.

ACKNOWLEDGEMENTS

The `JAMES` project is partially supported by *Agência de Inovação* and was accepted in the European Eureka Program (Σ!1921). Special thanks to Eduardo Lourenço and Pedro Pereira, for the development of the AgentX services, and to the rest of the project team.

REFERENCES

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

[2] "ISO/IEC 9595: Information technology - Open Systems Interconnection - Common management information Service definition", International Organization for Standardization, International Electrotechnical Commission, 1990

[3] G. Goldszmidt, Y. Yemini, "Decentralizing Control and Intelligence in Network Management", in Proceedings of the 4th ISINM, Santa Barbara, 1995

[4] J. Martin-Flatin, S. Znaty, "Annotated Typology of Distributed Network Management Paradigms", T/R SSC/1997/008, École Pol. Fédérale de Lausanne, 1997

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

[6] IBM Aglets Workbench, `http://www.trl.ibm.co.jp/aglets/`

[7] Mitsubishis' Concordia, `www.meitca.com/HSL/Projects/Concordia/`

[8] General Magic Odyssey, `http://www.genmagic.com/agents/`

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

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

[11] IKV++ Grasshopper, `http://www.ikv.de/products/grasshopper/`

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

[13] P. Simoes, L. Silva, F. Boavida, “Integrating SNMP into a Mobile Agent Infrastructure”, in Proceedings of DSOM’99, Zurich, 1999

[14] L. Silva, L. Almeida, “The Advantages of Using Mobile Agents in Software for Telecommunications”, in Proceedings of ICCC’99, Tokyo, 1999

[15] JavaSpaces, `http://java.sun.com/products/javaspaces`

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

[17] B. Wijnen, G. Carpenter, K. Curran, A. Sehgal, G. Waters, “Simple Network Management Protocol Distributed Protocol Interface Version 2.0”, RFC 1592, 1994

[18] G. Susilo, A. Bieszczad, B. Pagurek, “Infrastructure for Advanced Network Management based on Mobile Code”, in Proceedings of NOMS'98, New Orleans, 1998

[19] G. Goldszmidt, “Distributed Management by Delegation”, PhD thesis, pp. 84-87, Columbia University, 1996

[20] Java Dynamic Management Kit, `http//www.sun.com/software/java-dynamic`

[21] AdventNet SNMP, `www.adventnet.com/products/snmpbeans`

[22] “Mobile Agent System Interoperability Facilities Specification”, OMG TC Document orbos/97-10-05, 1998

[23] Foundation for Intelligent Physical Agents, `http://www.fipa.org`

[24] J. Nicklish, J. Quittek, A. Kind, S. Arao, “INCA: an Agent-based Network Control Architecture”, in Proceedings of IATA’98, Paris, July 1998

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

[26] A. Puliafito, O. Tomarchio, “Advanced Network Management Functionalities thorugh the use of Mobile Software Agents”, in Proceedings of IATA’99, Stockholm, 1999

[27] A. Sahai, C. Morin, “Enabling a Mobile Network manager (MNM) Through Mobile Agents”, in Proceedings of Mobile Agents’98, Stuttgart, Germany, 1998

[28] Bieszczad, A., “Advanced Network Management in the Network Management Perpetuum Mobile Procura Project”, Technical Report SCE-97-07, Systems and Computer Engineering, Carleton University, 1997