J.AGENTX: A TOOL FOR DYNAMIC DEPLOYMENT OF OPEN MANAGEMENT SERVICES

Luís Silva, Fernando Boavida

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

Abstract. *This paper presents J.AgentX, a Java-based toolkit for dynamic extension of SNMP agents. This toolkit provides an easy way to dynamically supply SNMP-based interfaces to new management services, reducing the costs associated with the deployment of these new services. J.AgentX is fully compliant with the recent AgentX standard (thus assuring interoperability) but goes one step further by including several features for faster service development, like a powerful high-level interface, simultaneous support for different communication mechanisms, a small footprint and portability.*

These features were successfully employed to provide SNMP-compliant network management services based on flexible and innovative delegation topologies. Two of those experiences will also be presented in this paper. In the first one J.AgentX provides an SNMP interface to management services based on mobile code. In the second experience J.AgentX is used to develop transparent proxies for legacy monolithic services.

KEYWORDS: Network Management, SNMP, Agent Extensibility

1. INTRODUCTION

The Simple Network Management Protocol (SNMP [1]) is often described as legacy and unattractive technology, unable to cope with the current management requirements. However, despite its well-known limitations and the relative success of competing technologies, like CORBA [2] and Web-based Management [3], SNMP is still the most widespread framework for network and systems management. For this reason, even new management services need to keep SNMP-support for integration with management platforms like HP Openview [4] and Tivoli [5].

The traditional SNMP architecture, however, fails to deliver many of the requirements of modern management services, like flexibility, dynamics, decentralization, and high-level functionality. This problem often results in a compromise: management services are built upon "better" technologies and made available through proprietary interfaces that lack integration; after that, an ad-hoc SNMP interface layer with poor functionality is added, in order to provide some degree of interoperability with legacy applications. This kind of compromise, if carefully handled, may result in powerful and technologically advanced management services that maintain support for legacy systems. However, the price tag for keeping legacy support is high: there is the cost of adding legacy interfaces and there is the functionality cut associated with their use. Those interfaces, if not well designed, result in inaccessible and unusable services.

In the process of designing good legacy interfaces there is one important technology: extensible SNMP agents. With dynamic extension of SNMP agents it is possible to separate the SNMP interface from the provided management services. An “empty agent” receives SNMP requests from the management applications and forwards them to registered “subagents”. Each subagent is responsible for a specific set of SNMP objects and knows how to translate incoming requests into management operations. Ideally, each hosted application or network service would dynamically provide its subagent, in order to become SNMP-manageable. In this way the SNMP agent is no longer monolithic, since it is possible to add support for arbitrary MIB (Management Information Base) modules not known at the beginning.

This technology brings other potential advantages, including easier development of SNMP management services — it is easier to develop a subagent than to create a full SNMP agent — and increased portability — most subagents are system-independent, unlike monolithic agents.

Traditionally, agent extension has been focused on the main problem (to overcome the limitations of monolithic systems) and the other potential advantages have been less explored. In our work we started from a different approach: we considered that the agent extension problem is already satisfactorily addressed by the current standards and we invested more on other topics, like easiness of development, simplicity, portability, flexibility and new areas of application.

This resulted in the development of `J.AGENTX` [6], a toolkit that constitutes the first Java-based implementation of the AgentX standard [7]. Unlike other tools, this lightweight toolkit is fully dedicated to agent extension and provides a high-level programming interface for faster development of new services. In order to exploit the technology in new application areas, several enhancements were also made at the communication level, with simultaneous support for different communication mechanisms.

This paper presents the `J.AGENTX` toolkit and proposes the application of dynamic SNMP agent extension to further decentralize the management process. Two specific experiences were made in this area: one with mobile code and another using proxy agents.

The rest of this paper is organized as follows: Section 2 is dedicated to agent extension technology, and Section 3 presents the `J.AgentX` package. Section 4 proposes the potential application of the `J.AgentX` framework into decentralized management. Section 5 concludes the paper.

2. DYNAMIC SNMP AGENT EXTENSION

Dynamic extension of SNMP agents is generally based on two entities: the master agent and the subagents. The master agent acts as the SNMP front-end, receiving SNMP requests from outside, forwarding them to the adequate subagents, and replying to those requests on behalf of subagents. Each subagent is specialized in providing a specific management functionality, "implementing" the MIB objects that represent that functionality. It registers those MIB objects in the master agent and receives forwarded requests related to those objects. After that, it processes those requests and replies to the master agent that will than reply to the outside application. This process is transparent to outside applications, since all they see is a single entity providing SNMP services — the master agent. Interaction between the master agent and subagents is based on a predefined "agent extension" protocol. Figure 1 shows this architecture.

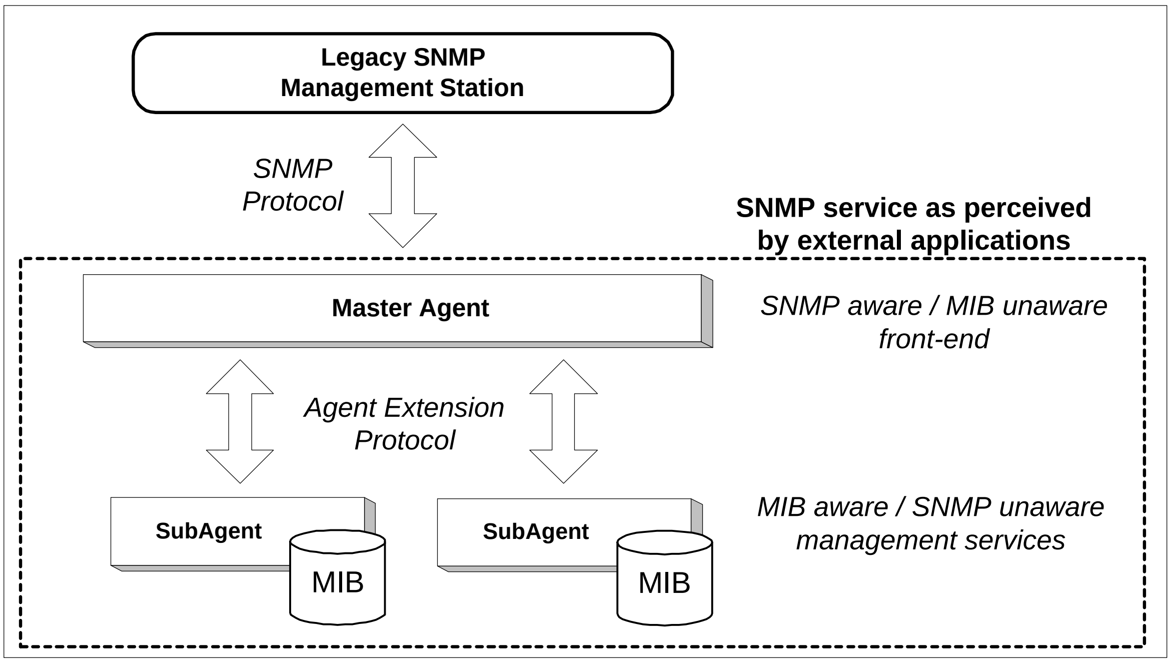
Legacy SNMP Management Station
SNMP Protocol
SNMP service as perceived by external applications
Master Agent
SNMP aware / MIB unaware front-end
Agent Extension Protocol
SubAgent
MIB
SubAgent
MIB
MIB aware / SNMP unaware management services

Figure 1 ¾ Typical Architecture of Extensible SNMP Agents

2.1 Protocols for Agent Extension

There are several proprietary solutions for agent extension, supposedly as an improvement of the available open protocols. These solutions claim several advantages, namely increased performance, greater flexibility and optimisation for specific hardware or operating systems. However, the minor technical gains provided from these proprietary solutions seldom justify the sacrifice of portability in the development of subagents.

Hardware and operating system independence, as well as interoperability between master agents and subagents from different vendors, is only achieved by using open protocols for agent extension. There are now three such protocols: SMUX [8], DPI [9] and AgentX [7].

SMUX (SNMP Multiplexer Protocol) is almost as old as SNMP itself. In SMUX communication with subagents (or peers, according to the SMUX terminology) is based on SNMP messages, resulting in several coordination and synchronization shortcomings. However, despite these limitations, SMUX is still the base of several commercial products.

DPI (SNMP Distributed Protocol Interface) took one step further, defining a specific protocol for interaction between the master agent and the subagents. In this protocol subagents no longer have to “speak” SNMP, resulting in simpler and smaller subagents.

AgentX (Agent eXtensibility Protocol) was proposed in 1998 as an evolution of DPI that enhances several technical aspects and handles different versions of the SNMP protocol. It recently reached the status of IETF standard specification and is expected to gradually replace all the other open and proprietary solutions.

2.2 AgentX Architecture and Implementations

The AgentX architecture is not much different from the general architecture described in Section 2. There is a master agent interfacing with SNMP applications but with little or no access to the management information, and there are the subagents, implementing the management services and the associated management information but with no direct contact with SNMP applications. The AgentX protocol provides mechanisms for subagent registration, for infrastructure administration, and for SNMP operations (including atomic sets involving several subagents). Communication is connection-oriented and TCP is the preferred protocol, although other options are possible.

The protocol also defines a specific AgentX MIB [10] for the management of the infrastructure itself. This is an elegant solution for remote management of the AgentX entities (e.g. monitoring subagents, verifying internal errors, close subagents) using SNMP.

There are now five known implementations of AgentX: three public-domain packages from the academic world and two commercial products.

The first public-domain implementation of AgentX was integrated in the CMU SNMP package [11] released in the end of 1998. It is a partial implementation that supports SNMPv1 and SNMPv2c and can accept AgentX connections via TCP or Unix-domain sockets. It includes a master agent — an evolution of the CMU SNMP Agent — and libraries to develop subagents.

The UCD-SNMP v4.1 package [12] implements AgentX in a rough way. It contains an SNMP Extensible Agent that supports SNMPv1, SNMPv2c and SNMPv3 but doesn't reference any support for the development of subagents.

`J.AGENTX` [6] is, to the best of our knowledge, the first Java-based implementation of the protocol. It includes a master agent, several libraries for the development of subagents, and an implementation of the AgentX MIB [10], for remote management of the infrastructure. `J.AGENTX` will be described in some detail in Section 3.

There are still few non-academic AgentX-compliant products. Compaq references support for AgentX in its Tru64 UNIX version 5.0 [13]. This support comprises the master agent, the subagent API library and the AgentX MIB. The Envoy SNMP Agent [14], from WindRiver, also claims AgentX compatibility for its proprietary protocol, named Envoy +X.

3. THE `J.AGENTX` PACKAGE

CMU-SNMP and UCD-SNMP are two traditional C-based, UNIX-oriented, general-purpose SNMP implementations with a wide spectrum of goals (agent development, application development, service development) that, over the years, accumulated several features. In this context, support for AgentX stands as just another feature.

The `J.AGENTX` package was designed from the scratch with a much stronger focus on AgentX and service development, resulting in simpler, smaller, and more elegant solutions. As already mentioned, the list of issues we took into account includes:

- Simplicity — the package should as simple to use as possible, excluding features with no relevance in the expected context of use.
- Portability — the package, as well as developed services, should be truly portable. For this reason `J.AGENTX` is 100% Java-based and system independent.
- Interoperability — the package should provide full interoperability with other AgentX implementations.
- Flexibility — the package should be able to adjust to different environments and different application fields. This means, for instance, the possibility of mapping AgentX into new communication protocols and the possibility of maintaining simultaneous support for several protocols.
- Easy service development — technical details of SNMP and AgentX should be truly hidden bellow high-level programming APIs, focusing the users attention on the services he/she is developing.

The result was a very compact package with two independent but complementary components (see Figure 2):

- a small, lightweight master agent;
- and an AgentX-API for development of AgentX subagents.

3.1 The `J.AGENTX` Master Agent

The master agent is a fully functional application that supports dynamic addition of sets of SNMP managed objects using the AgentX protocol. It is a small Java application that requires no further programming to be used.

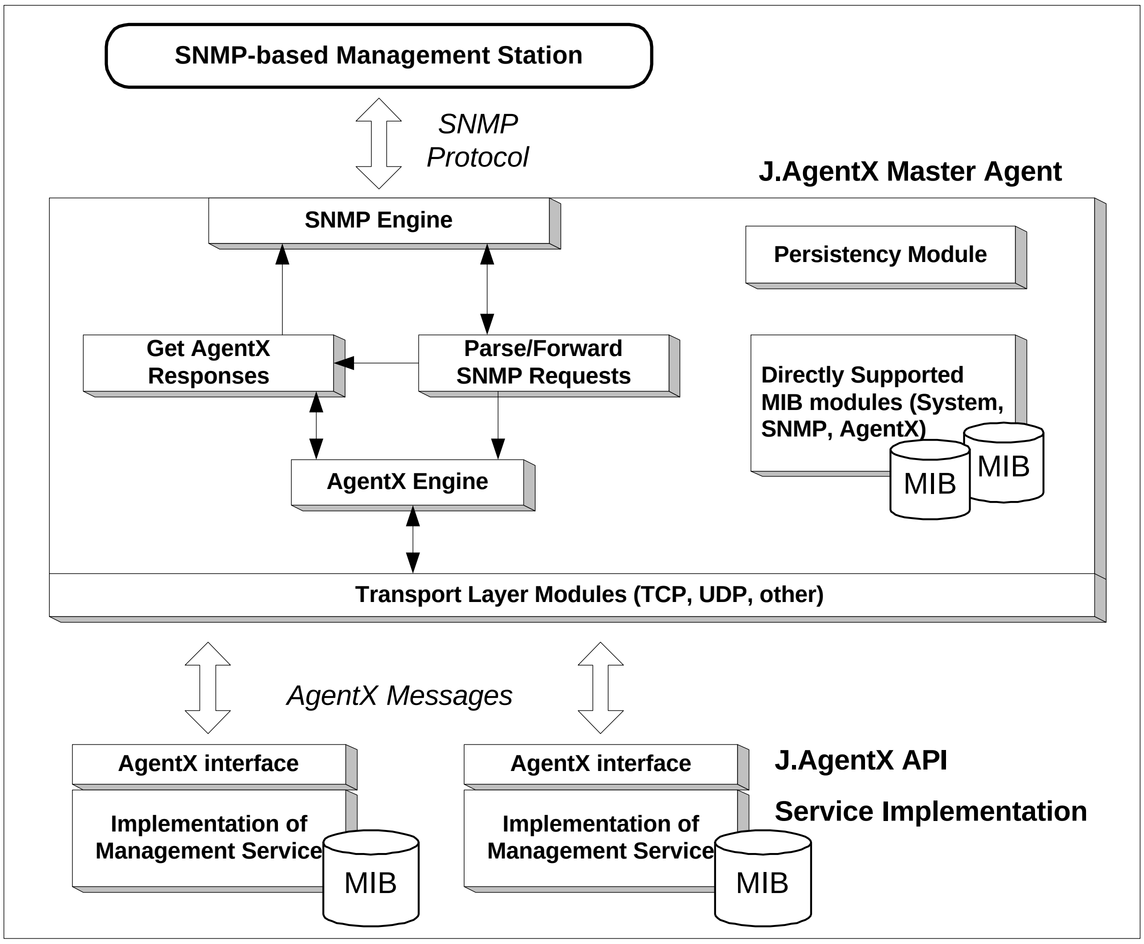
SNMP-based Management Station
SNMP Protocol
J.AgentX Master Agent
SNMP Engine
Persistency Module
Get AgentX Responses
Parse/Forward SNMP Requests
Directly Supported MIB modules (System, SNMP, AgentX)
MIB
MIB
AgentX Engine
Transport Layer Modules (TCP, UDP, other)
AgentX Messages
AgentX interface
AgentX interface
J.AgentX API
Service Implementation
Implementation of Management Service
Implementation of Management Service
MIB
MIB

Figure 2 ¾ Architecture of J.AgentX

According to early AgentX specifications, a master agent should be MIB unaware. However, there is management information directly related to this application. For this reason, the master agent directly implements, as an option, two groups of the standard MIB-II (System and SNMP) and the AgentX MIB. This solution does not compromise compactness, portability or interoperability, and is far more simple than keeping these MIB modules in external subagents.

Another important addition to the master agent was the simultaneous support for several communication channels[1], including UDP, TCP and application-specific solutions. With this addition it is possible to deploy AgentX-based services in environments with special communication requirements, like mobile code, without losing interoperability. With the modular design of the master agent the addition of new communication layers is straightforward.

3.2 The AgentX API

The AgentX API is a set of Java classes for easy development of AgentX subagents. It provides the programmer with an easy-to-use, high-level interface to the

[1] It should be stressed that, for non-technical reasons, the public-domain release only supports UDP and, in the near future, TCP.

extensibility protocol. A detailed description of this interface, which is outside the scope of this paper, may be found in [6], as well as some demonstrative examples.

4. APPLICATIONS OF `J.AGENTX`

Traditionally, agent extension mechanisms are used to develop flexible SNMP agents supporting dynamic addition of management services. This kind of application, that fragments management functionality through several specialized subagents, is indeed the primary application field for this technology.

However, as already mentioned, agent extension can also play a valuable role in more flexible and distributed management topologies. In this section we will present two simple experiences where we used `J.AGENTX` in non-conventional ways. In the first one management functionality was placed in mobile agents, resulting in services that dynamically change their location according to the circumstances. `J.AGENTX` was then used to provide an SNMP interface to those mobile services. In the second one we used `J.AGENTX` to develop proxies for integration of native management services.

4.1 Mobile SNMP-capable Services

Mobile agent technology (MAT) is one of the most recent and exciting approaches to distributed computing. 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 its state across the network. With this model, task processing can be dynamically distributed by the network and placed in the most appropriate locations, reducing the network traffic and increasing the scalability and flexibility of applications.

MAT is very useful for remote deployment of dynamic management services (constituted by one or several mobile agents). Using the features provided by MAT — like location transparent inter-agent coordination, mobility, disconnected operation, robustness, remote upgrading — it is possible to develop more powerful and flexible intermediate management services than with other distributed computing paradigms. However, these new services still need to interface with legacy applications.

The `JAMES` project [15] applies MAT in the development of applications for telecommunications and network management. A key component of this project is the development of a MA platform for the support of distributed management services. In this context, `J.AGENTX` plays an important role in the extensive `JAMES` framework for interoperability between MAT and the SNMP architecture [16].

In this framework a `J.AGENTX` master agent was placed in the MA platform to provide an SNMP front-end for legacy applications trying to access the platform or the hosted distributed services (performed by mobile agents). For increased flexibility (namely on-demand installation) the master agent itself consists of a mobile agent that can be instantiated according to the needs. For obvious reasons, however, this mobile agent will be stationary in a know location, instead of migrating across the network.

This front-end is then used by the mobile agents that implement the management services (see Figure 3). These mobile agents (or sets of coordinated mobile agents) act as AgentX subagents but their mobility is not sacrificed, since the AgentX messages are exchanged using the location-transparent inter-agent communication mechanisms provided by the platform, instead of plain TCP. With this architecture it is thus possible to create legacy interfaces for distributed management services that make full advantage of the MAT potential.

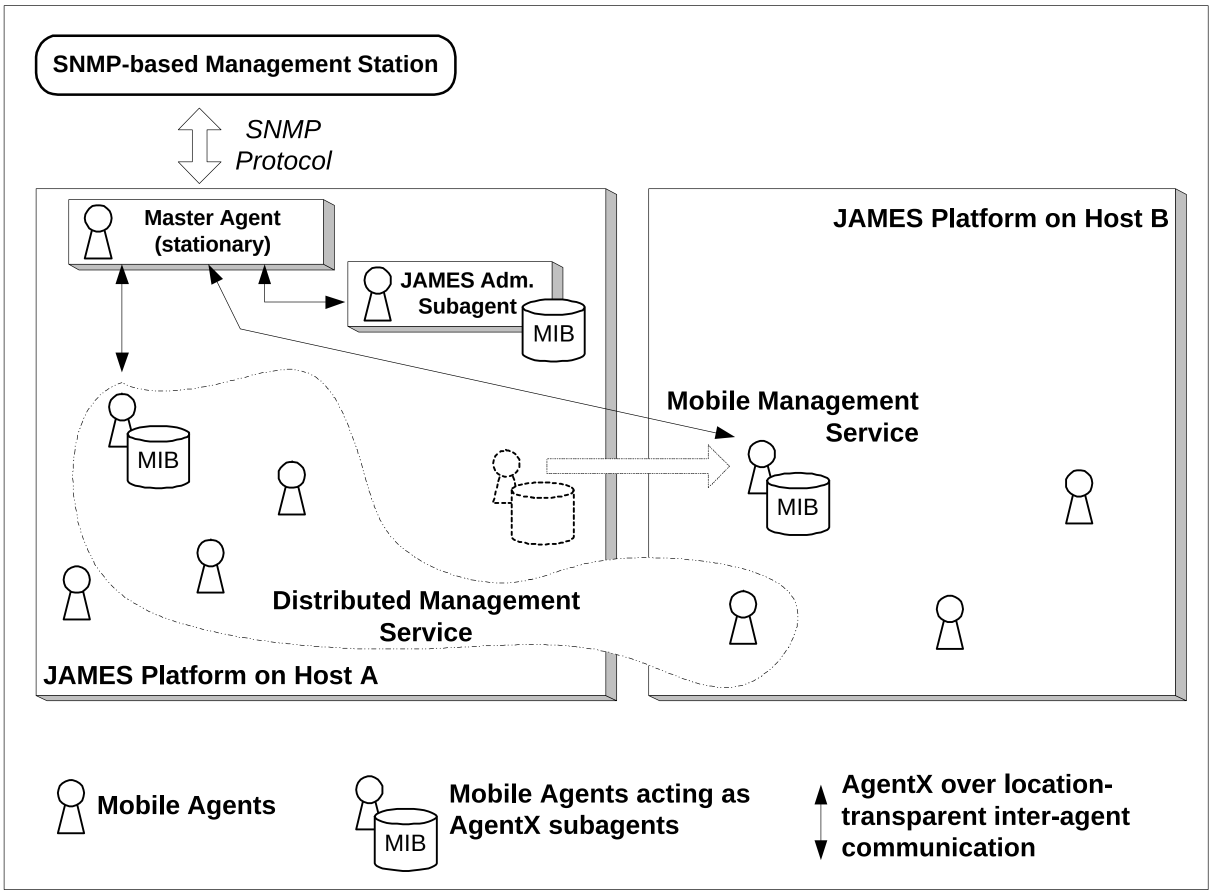
SNMP-based Management Station
SNMP Protocol
Master Agent (stationary)
JAMES Adm. Subagent
MIB
JAMES Platform on Host B
Mobile Management Service
MIB
MIB
Distributed Management Service
JAMES Platform on Host A
Mobile Agents
MIB
Mobile Agents acting as AgentX subagents
AgentX over location-transparent inter-agent communication

Figure 3 ¾ J.AGENTX in the Context of JAMES

A side benefit of this solution is the ability to provide an SNMP interface for the management of the JAMES platform itself. For this purpose another subagent was created to translate SNMP operations (over objects from a predefined MIB for administration of JAMES) into the internal platform management API.

This experience shows how two simple adjustments on the traditional applications of AgentX (the placement of the master agent on a mobile agent and the usage of location transparent communication mechanisms) are enough to provide legacy interfaces for mobile management services. A more detailed description of the JAMES integration framework for SNMP, that is outside the scope of this paper, is presented in [16-17]. An independent but similar approach to this subject, that also uses agent extension mechanisms to integrate MAT in the SNMP architecture, is described in [18].

4.2 Integration Proxies

A *proxy agent* is an SNMP agent that intermediates the communication between SNMP applications and non-SNMP management services. Proxies are usually hard-coded SNMP agents specifically designed to represent one legacy device.

This *proxy* concept, however, is not necessarily restricted to the representation of non-SNMP devices. Intermediate management services (i.e. a proxy agent) provide interesting decentralization solutions, even when SNMP-based low-level instrumentation is already available. In this context proxy agents may aggregate and pre-process low-level data from one or several managed entities in order to offer higher-level functionality to the central management stations, optimising performance, network traffic and scalability. This unorthodox application of the proxy concept is a possible solution to distribute management intelligence across the network — and closer to management data — while keeping legacy, centralized SNMP management applications.

Independently from the nature of underlying managed entities (SNMP-capable or not), with `J.AGENTX` it becomes possible to develop proxies as subagents, reducing development costs — it is no longer necessary to include an SNMP engine in the proxy — increasing portability and improving service integration.

Figure 4 presents a possible configuration for an extensible SNMP agent that includes two subagents acting as proxies.

The first is a “classic” proxy that provides SNMP functionality on behalf of a non-SNMP device. This corresponds to the classic usage of proxies.

The second proxy acts more like a middle layer service that autonomously handles the low-level management of a set of SNMP devices, providing an high-level management interface for upper management applications. This ad-hoc approach to management decentralization conforms with the SNMP architecture and requires no changes on the managed devices or on the management application.

The third subagent of Figure 4 is not related with proxy technology and shows how proxies can coexist with other subagents in order to provide an integrated management interface.

5. CONCLUSIONS AND FUTURE WORK

In this paper we describe `J.AGENTX`, the first Java-based implementation of the AgentX protocol for dynamic extension of SNMP agents. Unlike previous implementations, `J.AGENTX` is a small product that focuses on agent extension, not just an add-on to a large, general-purpose SNMP package.

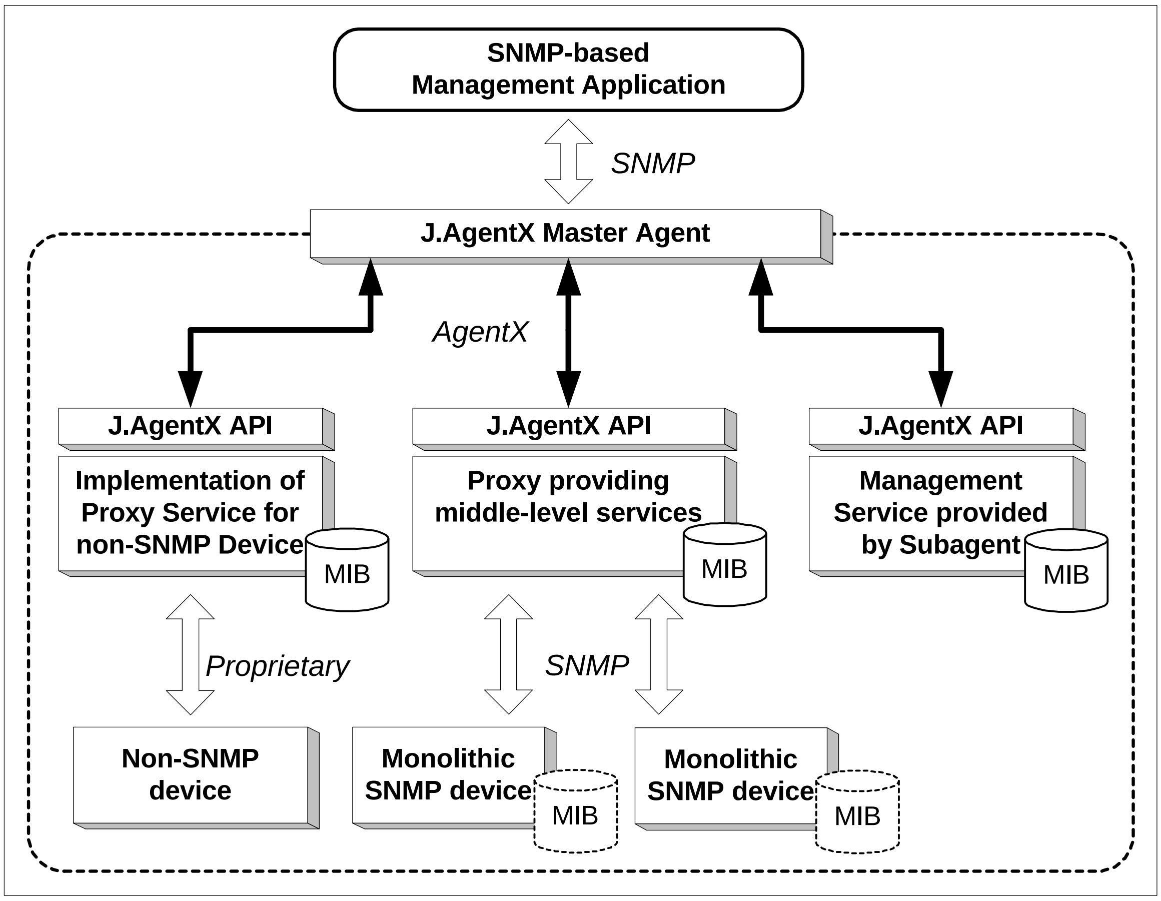
SNMP-based Management Application
SNMP
J.AgentX Master Agent
AgentX
J.AgentX API
Implementation of Proxy Service for non-SNMP Device
MIB
J.AgentX API
Proxy providing middle-level services
MIB
J.AgentX API
Management Service provided by Subagent
MIB
Proprietary
SNMP
Non-SNMP device
Monolithic SNMP device
MIB
Monolithic SNMP device
MIB

Figure 4 –Proxy Agents in the J.AgentX Architecture

Using agent extension technology it is possible to develop more flexible SNMP services, with clear separation between protocol issues (handled by master agents) and functionality implementation (provided by subagents). Portability and enhanced integration are other well-known advantages of this technology.

Agent extension technology, however, can also be used to provide ad-hoc decentralization within the SNMP architecture. In this paper we presented two such examples.

In the first one extensible agents were used to interface with a mobile agent infrastructure. Mobile agents are a powerful way to deploy new and advanced middle-layer management services, and agent extension acts as a bridge between these distributed services and the legacy SNMP-based management applications.

In the second example the concept of SNMP proxy was integrated in the J.AGENTX framework and further extended to encompass intermediate managers that autonomously handle the low-level management of SNMP-devices. This kind of ad-hoc delegation provides enhanced performance, flexibility and scalability without changes on the managed devices or on the management applications.

These two applications are not an intent to show that SNMP is appropriate for management delegation. It should be clear that SNMP is definitely not a good solution for decentralized management. However, for a number of reasons, it is often the only available solution.

ACKNOWLEDGEMENTS

Special thanks to Rodrigo Reis, for the development of the SNMP stack used by `J.AGENTX`.

The application described in Subsection 4.1 was developed in the context of the `JAMES` project, that is partially supported by *Agência de Inovação* and was accepted in the European Eureka Program (Σ!1921).

REFERENCES

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

[2] OMG, "The Common Object Request Broker Architecture and Specification, 1995

[3] Webm

[4] Hewllet-Packard Openview, `http://www.openview.hp.com`

[5] Tivoli, `http://www.tivoli.com`

[6] University of Coimbra, The J.AgentX Package, `http://www.dei.uc.pt/agentx`

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

[8] M. Rose, "SNMP MUX Protocol and MIB", RFC 1227, 1991

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

[10] L. Heintz, S. Gudur, M. Ellison, "Definitions of Managed Objects for Extensible SNMP Agents", RFC 2742, 2000

[11] The Carnegie Mellon University AgentX Implementation, `http://www.net.cmu.edu/groups/netdev/agentx`

[12] University of California - Davis, UCD-SNMP, `http://ucd-snmp.ucdavis.edu/`

[13] The Compaq Tru64 UNIX Product, `http://www.tru64unix.compaq.com/`

[14] Wind River Networking Products - Envoy SNMP Agent, `http://www.wrs.com/products/html/envoy.html/`

[15] 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, Springer-Verlag

[16] P. Simões, L. Silva, F. Boavida, "Integrating SNMP into a Mobile Agent Infrastructure", in Proceedings of DSOM'99, Zurich, 1999

[17] P. Simões, R. Reis, L. Silva, F. Boavida, "Enabling Mobile Agent Technology for Legacy Network Management Frameworks", Proceedings of SoftCOM'99, Split, 1999

[18] CARLETON, NOMS2000