Network Management Using Mobile Agents

Jorge F. A. da Costa, Sergio V. Fialho
Universidade Federal do Rio Grande do Norte, Natal RN, Brazil

Abstract — **This paper proposes a mobile agents based framework to implement the management of computer networks. The traditional approach used by SNMP is based on a highly centralized topology. For complex network computers, such a topology may cause overload conditions in the use of network resources during management activities. The adoption of mobile agents can provide for the lack of a manager-manager communication mechanism in the SNMP environment. A basic environment using the Aglets Workbench was developed, where a Mobile Agent was capable of carrying the SNMP management software and install itself in previously chosen machines in a net. Some tests were performed and they confirmed the viability and convenience of the proposed solution. Starting from the implemented framework, it is possible to build a complete system for Computers Network Management.**

I. INTRODUCTION

The most widespread technology for computer network management uses the SNMP protocol. Its currently used version relies on a centralized management topology, meaning that only one station concentrates all the managed information produced by hubs, switches, routers and even servers and workstations. The newest versions of SNMP foresee a distributed management framework, introducing manager-to-manager communication capabilities. However, none of those protocol versions are available in any of the several devices and services produced at the moment. So, the initial scheme of a highly centralized network management framework is maintained.

An alternative foreseen to implement remote management in the context of the present version of SNMP is the RMON MIB. It uses probes, installed in LAN segments, to obtain information from SNMP agents, before sending it to the central manager station.

This work proposes a more thorough alternative to the use of that technology, and demonstrates the viability and convenience in the use of mobile agents for network management purposes. These agents can also be provided with AI techniques to implement advanced network and computer management capabilities. In the following it is presented some of the advantages of this approach:

- the network does not become overloaded, because the management activity is implemented in a distributed way [1], by means of several managers installed by the mobile agents in equipments spread throughout the net. These agents can recover the locally monitored information, filter them and return to the central manager node just what really matters.
- it doesn't depend on the installed hardware/software, such as the RMON MIB
- it may be easily updated because it is based on software
- the control can be centralized in any point of the net where the system is installed.

In the next section, it is introduced the mobile agents concept, showing some of the most used agents platforms. The Aglets platform will be described in more detail, along with its main characteristics and advantages. In section III, the Agent Based Management and its associated execution environment are introduced. The results obtained are described in the fourth section. Finally, in section V, the major conclusions and perspectives for future works are discussed.

II. MOBILE AGENTS

Agents are software programmed to accomplish certain tasks without users intervention. According to Bieszczad [2], mobile agents are self-contained software pieces that are capable to autonomously migrate inside a net. In this paper, a manager-agent interaction model, similar to the client-server model and shown in Fig. 1, was used to depict the dynamics adopted by agents in its environment. In this model, there is a program that performs the manager function of coordinating the agents work. On the other hand, the agents execute the tasks cooperating with each other and with the manager. The latter is responsible for collecting and processing the data received by the agents, acting as a server.

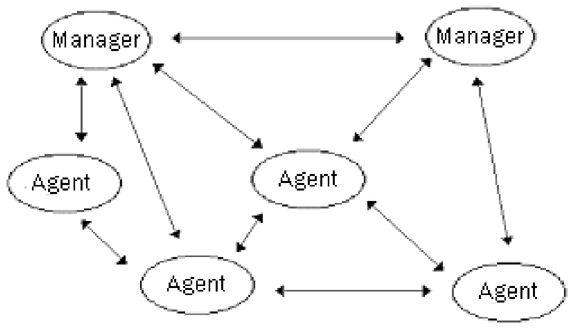


Fig. 1. The Manager-Agent Cooperation Model

Jorge F. A. da Costa is with Departamento de Engenharia Elétrica and Sergio V. Fialho is with Departamento de Engenharia de Computação e Automação, Universidade Federal do Rio Grande do Norte, Natal-RN, Brazil, 59072-970. PABX: +55 (84) 215-3999.
E-mails: flavio@cdiauto.com.br, fialho@pop-rn.rnp.br.

An agent migrates in a distributed environment from an agency to another. According to [3], an agency represents a “logical” place in the distributed system. When an agent migrates, its execution is suspended in the original agency and the agent is transported (i.e., code, data and execution state) to another agency where the execution is continued.

Some of the possible applications for the mobile agents are:

- Searching of information in the Web or in a database
- Parallel Processing of scientific data
- Variable data supervision applied to control purposes
- Network management

Techniques of Artificial Intelligence can be embedded in an agent to improve its autonomy degree [4], so that the agent can react to the environment conditions and to learn from them.

In the next sub-sections, some issues on mobile agents execution and its main characteristics are shown. Aglets Workbench (AWB) will be presented in more detail, because it was the platform chosen for the development of this work. A more complete and current list of mobile code systems can be found in [5].

A. Odissey

Developed by General Magic, it is considered the successor of Telescript. Odissey [6] is constituted by a group of Java libraries and it uses RMI (Remote Method Invocation) as its transport and communication interface. The Odissey security is limited to the one provided by Java. However, General Magic plans to incorporate some Telescript security mechanisms into Odissey.

B. D’Agent

Developed in Dartmouth College, Agent TCL is based on TCL, that is an interpreted script language initially developed to Unix systems. It was ported to another languages, such like Python, Scheme and Java, and changed its name to D’Agent [7]. When traveling from a host to another one, a pro-active migration mechanism allows the transfer of the D’Agent command shell also. The security implemented in D’Agents consists of the possibility to encrypt the agent that is being transferred, by means of Pretty Good Privacy (PGP) or of the SafeTCL language.

C. Agents for Remote Access (ARA)

According to [8], ARA is a platform for agents able to move freely and easily without interfering with their execution, utilizing various existing programming languages and existing programs, independent of the operating systems of the participating machines. As seen above, ARA isn’t really an execution environment for agents but a pattern in which we can build our agents. It provides the system-level facilities to execute and move programs, let them interact and access their host system, all in a portable and secure manner.

D. TACOMA

TACOMA (Tromso And COrnell Mobile Agents) [9] is similar to D’Agents, because it too is an extension of the TCL language to provide support for mobile code systems. It supports agents applications developed in several languages, like Perl, Phyton, Scheme and C. Security in TACOMA is based on two different mechanisms: a list of reliable hosts, where agents will be accepted from, and the confinement of the agent execution to a reserved part of the machine file system.

E. Aglets Workbench (AWB)

AWB is a platform for agents development [10] totally written in Java and developed by the TRL Group, a subsidiary of IBM Japan. It possesses as main characteristics the development based on patterns, that is, models of pre-written classes from which is possible to extend and to create a new agent fast and easily. AWB is a Open-Source tool and its API was approved by a standardization committee.

The platform independence is one of its merits. The agents can run in any machine that implements a Java Virtual Machine, not mattering what is the installed operating system, allowing a great exchange of agents over several platforms.

The communication among the agents is processed through a message passing mechanism [11]. The messages can contain any object type allowed by the Java language, from a data file to a class containing a new Aglet that may be transmitted along the net.

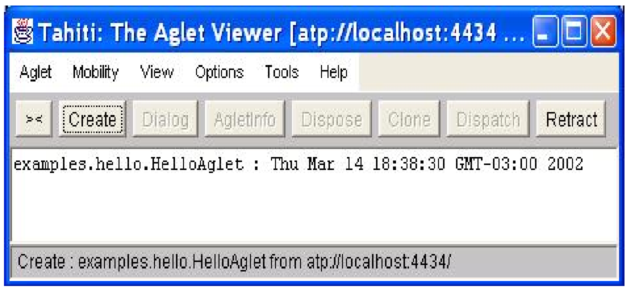
AWB implements the agents mobility by means of the Aglets Transfer Protocol (ATP), based on RMI, and in the objects serialization, both native resources in Java. The serialization allows the translation of the heap image of an Aglet in a sequence of bytes. The stack and the instructions counter of the Aglet threads cannot be serialized. That limitation happens, due to the Java Virtual Machine (JVM) architecture, because it doesn't allow the access and the direct manipulation of the program stack. That is an intrinsic feature of Java aiming at the reinforcement of the language safety. That is to say, when an Aglet travels from a host to another, its data will be maintained but its execution state will return to the beginning.

Included in the AWB package, there is the Tahiti Server that is responsible for the agents authentication and management. Tahiti will be described in the following.

E.1 Tahiti Server

The agents developed for AWB need a manager ambient in which they can be created. This environment is known as the Tahiti Server which implements a graphic interface for the AWB Daemon class. As it can be seen in Fig. 2, Tahiti allows to create, clone, dispatch and to destroy agents, all in an integrated execution environment. Still, it is possible to obtain information on the agents parameters (author's name, contact e-mail, and other) and bring it back to the origin host.

A demand to execute the aglets in an external ambient, like Tahiti Server, comes from the need of implementing a security politics that implements support for cryptography and authentication, capable of denying host access to agents that doesn't own recognized credentials. There are two categories of agents: the trusted (reliable) and the untrusted (unreliable) ones. One can define safety levels desired for each category. The permission levels include from hard disk access, stepping through access to certain host internal ports and going to access to the external net. All the items can be configured individually and in different categories.



```
examples.hello.HelloAglet : Thu Mar 14 18:38:30 GMT-03:00 2002
```

```
Create : examples.hello.HelloAglet from atp://localhost:4434/
```

Fig. 2. Tahiti Server running HelloAglet

III. AGENT BASED MANAGEMENT

As an alternative to a centralized management model, a framework for distributed management using mobile agents is proposed herein. In this framework, agents follow the Master-Slave paradigm with bi-directional communication mechanisms between them. The chosen system for agents deployment was the Aglets Workbench. In the following, the main reasons for this choice are presented:

- it is a mature architecture and takes advantage of the extensibility and reusability features of the Java language,
- it uses the native JVM Security Manager allowing a fine-tune adjustment of the permission rules,
- it adopts the open-source development model supplying the opportunity to alter the source code in agreement with the project needs,
- it is possible to easily implement an WEB interface for the communication to Tahiti Server.

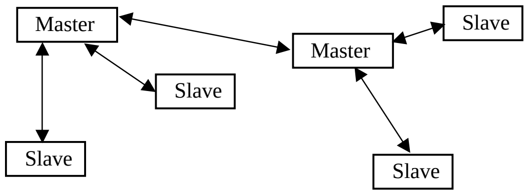


Fig. 3. Communication Scheme in the Master-Slave pattern

The Fig. 3 shows that the message exchange between Master and Slave agents occurs in a slightly different way from what was initially presented in Fig. 1. Now, the Slave agents communicate only with their respective Master, and the Masters can change messages with each other. This approach was used because it facilitates the coordination and the delegation of tasks, by allowing a better balance of the necessary amount of agents present in the net.

In small nets, just one Master agent is sufficient. As more nodes and sub-nets are added to the backbone, new Master agents may become necessary and can be introduced in the net. The main Manager Master may perform a cloning of itself and install it in another host, that will become responsible for the management of that specific sub-net. This feature offers high scalability allied to a good performance, because the agents are not required to carry great amounts of code or use a lot of memory, making it a good example of the use of the "divide to conquer" paradigm.

The SNMP interface chosen for the agents development was the SCK (SNMP Construction Kit) [12] library. That package owns a compact API (Application Programming Interface), which is easy to understand and is adherent to the open code philosophy. The SCK library implements the basic GET and SET SNMPv1 operations, defined in RFC 1157, besides of supplying Java Beans components for the construction of user friendly Graphic Interfaces.

IV. THE MANAGER AGLET

To exemplify the viability of the proposed framework, a Manager aglet was implemented. It will use a Manager Slave aglet in a nested topology, according to the Master-Slave hierarchy. The Manager Master sends the Manager Slave aglet to a remote host. When arriving at its destiny, the Slave Aglet becomes responsible for the management of the SNMP agents situated in its actuation area. This Manager Aglet example was based on the FingerAglet, included in the AWB package. Its functionality was extended by adding the SCK library, making it capable to monitor SNMP agents.

For simplicity reasons and simulation purposes, the whole test environment was reduced to just one only machine. The machine operating system was Windows 2000, running Tahiti Server and the SNMP service. When the Manager Master dispatches the Manager Slave, the destination address supplied is the loopback IP address 127.0.0.1. This will cause the return of the Slave aglet to the same system. That aglet will then perform the SNMP queries on the same IP address (local machine) which is running the SNMP agent. When job is finished, the Manager Slave goes back home and delivers the retrieved data to the Manager Master. The Java Classes used in the Manager Aglet implementation are described in the following.

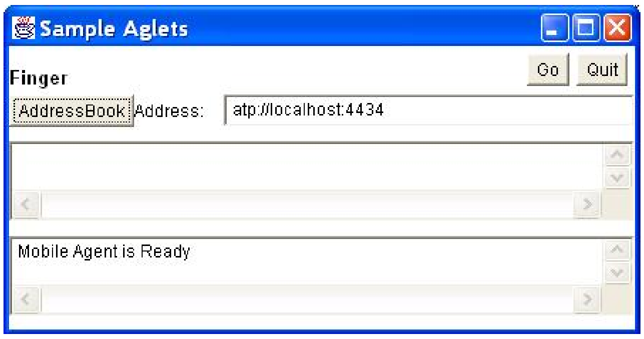
A. Manager Class

This Class is used to retrieve local user information from a remote aglet server (Tahiti). This information includes the name, organization, email address, and the local time at the remote server.

Given an ARL (Agler Resource Locator) it will dispatch a slave (an object of the ManagerSlave Class) to retrieve local user information. The slave will return with the collected information ready to be displayed by the master.

B. ManagerWindow Class

This Class produces a main window for user interaction. Through it, the human user may choose the destination address to where the ManagerSlave Aglet must be sent, as shown in Fig. 4.



```
Address: atp://localhost:4434
Mobile Agent is Ready
```

Fig. 4. The initial Program Window

It is possible to recall the addresses stored in the Tahiti Server AddressBook. The Aglet status is shown in the bottom Text Area. The Go button is used to dispatch the ManagerSlave aglet and the Quit button terminates the program.

C. ManagerInfo Class

The ManagerInfo class encapsulates the data of interest inside a Java Object, which contains several data Strings (chains of characters). The ManagerSlave uses ManagerInfo to deliver data to the Master Aglet.

This class is transported in the form of a Java object, translated into bytecodes, by the AWB message passing mechanism. This procedure increases security, since data is not transported in plain text format, as it does with the SNMP protocol, hindering the action of sniffers infiltrated in the net.

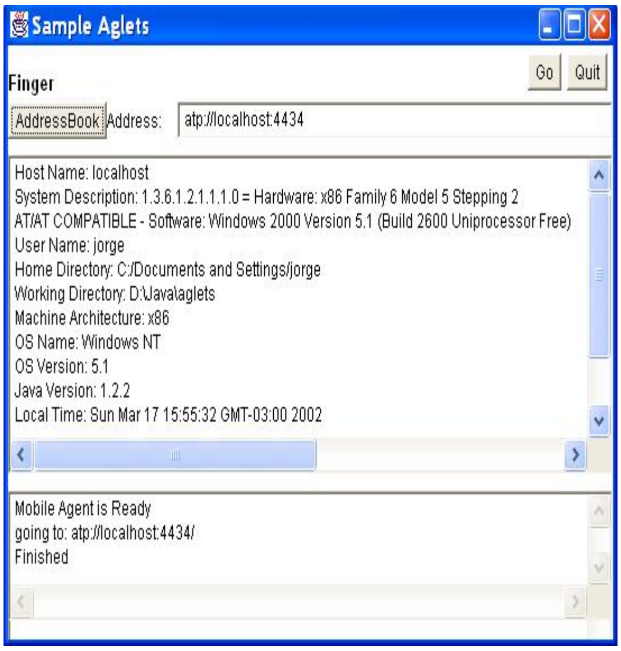
D. ManagerSlave Class

Once installed at its destiny, the ManagerSlave Aglet performs a query on the system properties, supplied by JVM, such as the logged user name, the operating system version and the host local time.

Similarly, the ManagerSlave Aglet makes a SNMP query on the MIB objects, for instance the object 1.3.6.1.2.1.1.1.0, which stores the system description (sysDescr) in ASCII format. After retrieving all those information, ManagerSlave Aglet encapsulates everything in the FingerInfo object and returns to its Master server. When arriving there, it delivers the retrieved information to its Master Aglet, by means of a message that contains the ManagerInfo object. The Master Agent processes that message and exhibits the result on the Main Window (ManagerWindow), as shown in Fig. 5.

The top Text Area exhibits the data delivered by the ManagerSlave to the Manager Aglet, in a user friendly way. The System Description field was obtained from the SNMP agent, and the other data was taken from the JVM's System Properties, by the ManagerSlave aglet.

The tracking information is found in the bottom Text Area, where is possible to follow the Slave Aglet steps and its life cycle.



```
Address: atp://localhost:4434
Host Name: localhost
System Description: 1.3.6.1.2.1.1.1.0 = Hardware: x86 Family 6 Model 5 Stepping 2
AT/AT COMPATIBLE - Software: Windows 2000 Version 5.1 (Build 2600 Uniprocessor Free)
User Name: jorge
Home Directory: C:/Documents and Settings/jorge
Working Directory: D:\Java\aglets
Machine Architecture: x86
OS Name: Windows NT
OS Version: 5.1
Java Version: 1.2.2
Local Time: Sun Mar 17 15:55:32 GMT-03:00 2002

Mobile Agent is Ready
going to: atp://localhost:4434/
Finished
```

Fig. 5. Result of a Manager Aglet consultation

V. CONCLUSIONS

It was demonstrated herein the possibility to accomplish a distributed network management by using mobile agents. A Manager Agent was developed to travel to a certain host and collect local information, including those made available by the SNMP MIB in that host.

One may improve the agents autonomy, by adding some techniques of Artificial Intelligence, embedding cooperative communication mechanisms in them. This will lead to an intelligent distributed management system.

Indeed, this work is being continued and presently it is under study which AI techniques are the best suited to be encapsulated in the mobile agents. Also, it is under development an improved version of the user interface, in order to display management data in a more friendly way.

The distributed management model will eventually supplant the current centralized model along the next years. As computer networks keep growing, with more and more terminal equipments and handhelds running TCP/IP being added to them, sometimes in a dynamic way, it becomes necessary the installation of a robust management system, capable of dealing with those factors. The Distributed Management framework using Mobile Agents fits most of these needs and requirements, still, it allows a greater flexibility, because the Agents can be endowed with intelligence and be able to adapt themselves to the environment conditions.

REFERENCES

[1] O'Malley, Scott A. and DeLoach, Scott A. *Determining When to Use an Agent-Oriented Software Engineering Paradigm*, Proceedings of the Second International Workshop On Agent-Oriented Software Engineering (AOSE-2001), Montreal, Canada, 2001.

[2] Bieszczad, A.; Pagurek, B.; White, T. *Mobile Agents for Network Management*, IEEE Communications Survey. Available at http://www.comsoc.org/pubs/surveys, 1998.

[3] Cockayne, William R. and Zyda, Michael, *Mobile Agents*, *Manning Publications Co.,* vol. 1, 1998.

[4] Khosla, R. And Dillon, T. *Engineering Intelligent Hybrid Multi-Agent Systems*. Kluwer Academic Publishers, 1997.

[5] *Mobile Agent List*. Available at http://www.informatik.unistutt-gart.de/ipvr/vs/projekte/mole/mal/preview/preview.html

[6] GenMagic Inc. *Odissey Home Page*. http://www.genmagic.com /technology/odissey.html

[7] Gray, R. *Agent TCL: a transportable agent system*. CIKM Workshop On Intelligent Information Agents, 1995.

[8] Peine, Holger and Stolpmann, Torsten. *The Architecture of the Ara Platform for Mobile Agents*. Available at http://citeseer.nj.nec.com/peine97architecture.html, 1997.

[9] Johansen, D.; Van Renesse, R.; Schneider, F. B. *An Introduction to the TACOMA Distributed System.* Technical Report 95-23, Tromso and Cornell University, 1995.

[10] TRL Group, *Aglets faq*. Available at www.trl.ibm.com /aglets/faq.html. IBM Corporation.

[11] Oshima, M; Karjoth, G. and Kouichi, O. *Aglets Specification 1.1*. Available at www.trl.ibm.com/aglets/spec11.html. IBM Corporation, 1998.

[12] Soun, Y. *SCK (SNMP Construction Kit)*. Available at http://membres.lycos.fr/ysoun.