A Mobile Agent Manager

Rui Pedro Lopes[1], José Luis Oliveira[2]

1. Polytechnic Institute of Bragança, ESTiG, 5300 Bragança, Portugal
rlopes@ipb.pt
2. University of Aveiro, , DET, 3810 Aveiro, Portugal
jlo@det.ua.pt

Abstract: Friendliness on user interfaces has been mostly forgot in mobile agent research. In fact the multitude of problems that have to be solved have left this more "simple" piece of the package out of target. However, the success of technology comes also from this important layer in the product development.
This paper describes an application that integrates monitoring data from the mobile agents environment, process and presents information in different ways according to different user requirements and allows the user to remotely control mobile agents entities.

1 Introduction

A mobile agent (MA) is a piece of software that is able to migrate from a particular environment to some other remote site where it can execute a specific function (system configuration, information gathering, ...) and then migrate to other systems in order to continue its job [1].

To accomplish this scenario, mobile agents do require runtime platforms to provide them with the necessary resources they need to operate and to migrate. Besides, this support system must also handle agents' life-cycle, security issues and, some times, additional services like multi-agent communication or coordination. This infrastructure is known as the agent system (AS) or agency, according to a well accepted nomenclature, and it is developed as an operating system extension.

From a more conceptual view, the typical role of a mobile agent is to perform an action in the representation of some entity, person, service or agent. The definition of its role is typically embedded in the agent code but its itinerary can be statically or dynamically defined by the owner/user or, using its own initiative and according to collected data, it can infer the following movement. During the mobile agents navigation there are several problems that can arise related to communication, access control, agents rights, agencies accessibility, just to name a few. In this context, the user should be able to monitor and control what they are doing, individually or as a community, and to change its behavior whenever it is necessary.

One of the problems with current mobile agent systems is the considerable difficulty in maintaining a large, distributed and remote infrastructure. The absence of standards for agent systems APIs (Application Programming Interfaces) limits or even eliminates the possibility for interoperability between different vendors' products. This also means that the tools available for monitoring and management are proprietary. However, it is important to monitor and control the infrastructure to conduct performance evaluation and performance tunning, fault

recovery/diagnostics, early detection of error or lapses on the overall service and load-balancing [2].

As agents work all over the network, the user must know what is going on globally and what is the impact on the network and on the agencies. He should be provided with information such as:

- What is the navigation plan of an agent? From where did it come and to where is it going?
- What is it doing?
- Where are the agencies geographically located?
- Which one is the most visited agency on the network?
- What is the CPU usage on every agency?
- How much time does this agent spends in each agency?
- What is the relation between the agency load and the number of mobile agents currently visiting it?

The user interface provided by some agent system (AS) is usually based on the File Manager paradigm which is insufficient to transmit an adequate and rich idea about the state of the mobile agents as individuals or as a community, the relations among them and the impact that they are causing in the network resources. Such paradigm is useful for structuring static information and for accessing hierarchical structures that do not change frequently, such as region information – the mobile agents directory service, agent systems and places –, but they cannot cope with several and very dynamic parameters. It may be adequate to represent the entities inside the AS, however how could this tree-like view show agents appearing and disappearing as they are created and destroyed? How could it show where migrations happens the most? Moreover, migrations would not show up until the user opens a branch (by clicking on the '+' sign for instance). It is also very difficult, if not impossible, to represent past and future information (where did this mobile agent come from? Where is it going to?).

The research around MA have not taken to much attention on the user interface. In fact, most of the past problems come from technical operational constraints and from the search for adequate application scenarios. Even the agent systems interoperability has not been too important on the last years research. However, mobile agents already have a role in the computing and communication market and the credibility of this technology will increase as solutions for interoperability, security assurance and friendly management will appear.

This paper presents a visual-oriented manager that helps to control mobile agents execution and navigation across multiples agencies and regions.

In order to present information in an intuitive user interface it is necessary to retrieve data from the agent system. Good sampling strategies and a well planned information system are necessary to avoid overcharge of the network bandwidth and computational resources. This subject will be detailed in section 2. In section 3 we describe a graphical user interface that can represent geographical information together with MAs and agent systems working parameters. Its main characteristic is the possibility to show, in an integrated view, the overall system operation. Finally, in section 4 we present some implementation details, followed by some conclusions.

2 Mobile Agents Information System

The information system behind the operation of a mobile agent infrastructure is structured in two main parts. One is related to the services provided by the agents, such as the operations they perform, the itinerary they have or their current state; the other is related to the parameters that represent behavior, such as “where they are” or “what are they doing”. From the user perspective, the former is meaningful for defining mobile agent based operations and the later is useful for infrastructure management purposes.

When agents are migrating and therefore changing their location repeatedly, the user may have more difficulty in maintaining an updated and accurate knowledge about where they are and what they are doing. The task of monitoring a distributed environment is more complex than the task of monitoring a single host or application. The number of AS can easily rise to dozens or hundreds and the performance of the systems depends on several factors including the network throughput and topology. It is important not to overload the network with the sampling mechanism but it is also important to avoid loosing resolution by using an insufficient sampling frequency [3].

The information system should be as generic as possible to allow using a broad set of mobile agent systems, regardless of the technology and vendor. The Mobile Agent Facilities (MAF) of the Object Management Group (OMG) provides such generalization and is a good starting point [4].

2.1 Information type

The user has access to the mobile agent infrastructure by retrieving and sending information. The information allows him to start operations as well as to check on how things are going. To design the information system we started the user requirements by enumerating the operations that should be available to the user.

The first condition was the integration of facilities of agent systems coming from different vendors. The user should be able to start jobs or tasks with specific paths or services. This means that the user should be able to create mobile agents and to provide them with appropriate arguments – even if the operational constraints of an agent framework avoids its agents to run in different systems (it will be confined, in this case, to the agencies of the same kind). He should also be able to monitor the agent location and to suspend, resume or terminate the agent operation.

Tracing and changing the agent's path in real time is also an important feature. This allows the user to keep a record of past and foreseen movements. To maintain this information accurate, it is necessary a) some notification mechanism or b) continuous polling on the agent location. The later may require large bandwidth to work properly. If the number of migrations per second between two agencies, which we call the connectivity between agencies, is high, this means that both agencies play an important role for the operation being performed by the mobile agents or that something is wrong with the paths they should be walking.

There are also some other useful parameters to evaluate the behavior of the agents. These are the CPU load of the host where the agencies are located, the number of mobile agents currently sharing an agency and the number of static agents. These values should be possible to correlate, to make it possible to take

decisions on the capacity of the host to support the number of agents to the available processing power. It is also important to store historical information about these parameters so that we can present to the user their evolution and to compare past with current values.

MAF is a collection of definitions and interfaces that provides generic access to mobile agent systems. It uses OMG IDL to declare two interfaces, which define all the operations on the agent system and on the region: the `MAFAgentSystem` and the `MAFFinder`. They can be used to provide vendor and platform independence [5].

Table 1presents a summary of the information system as well as the available methods to get/set the information.

Table 1. Summary of the operations associated with the information system

User Operation	Available method
Create agents	MAFAgentSystem::create_agent()
Monitor agent location	MAFFinder::lookup_agent()
Suspend, resume agents	MAFAgentSystem::suspend_agent()
Resume agents	MAFAgentSystem::resume_agent()
Terminate agents	MAFAgentSystem::terminate_agent()
Trace agent movement	Callback, notification or event mechanism (Proprietary)
Explicitly change the agent path	The user must change the agent state and desire (Proprietary)
Agency connectivity	Results of the ratio between migration tracing and time
CPU load	Operating system query (Proprietary)
Number of MA in the agency	MAFAgentSystem::list_all_agents()
Number of static agents	MAFAgentSystem::list_all_agents()

In addition to changing values, there are also several sampling operations that must be performed periodically to build the information system. The next section focuses on this subject in grater detail.

2.2 Sampling and changing

For the agent infrastructure to be contacted, either for retrieving values or for changing parameters, it is necessary to provide a common mechanism that both parties share. Moreover, it was decided that the mechanism should be independent of the agent system technology and vendor, perform automatic measurements and provide a minimal impact on network and computational resources. A further desired feature is that the sampling could be performed periodically or on demand based on a specific event. This requires time stamping every measurement so that it can be appropriately correlated with past and with related measurements.

The communication depends on two characteristics: the syntax – the way information is transported, and the semantics – the meaning of the information. Agent system communication is usually performed through an API which allows

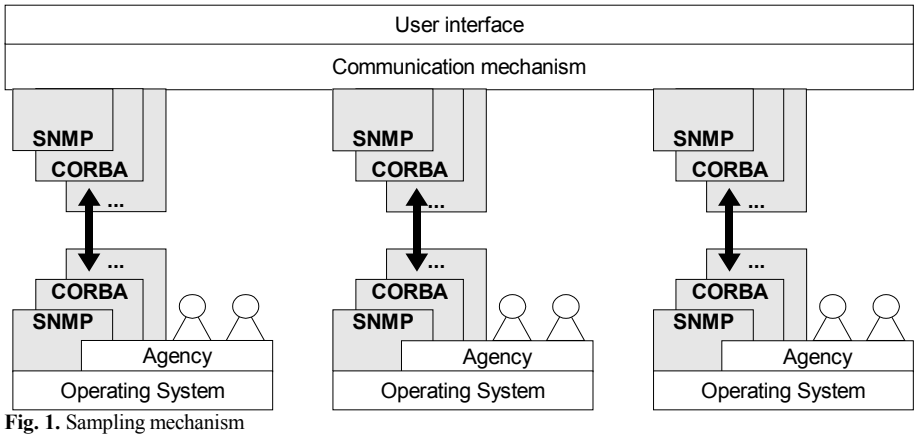

Fig. 1. Sampling mechanism

remote, high level access to the system parameters. This API is seldom standard but there are several standard protocols and models that can be used, such as SNMP (Simple Network Management Protocol) [6][7], CORBA [4] or other, since the agent system supports them (Fig. 1).

The communication mechanism must follow the syntax defined by the agent system. Semantics has to do with the meaning of the information and implies some kind of processing. Some parameters are usually more meaningful if processed before being consumed. For example, the number of migrations between two agencies is not meaningful unless associated with time information. Saying that 2312 migrations occurred between two agencies is not meaningful until we say "in the last 20 seconds". Other parameters may be meaningful by themselves, such as processor load or the number of mobile agents currently parked at an agency. This makes us foresee two kinds of sampling: absolute – the sampling value corresponds to the retrieved value – and delta – the sampling value corresponds to the difference between the current and the previous values.

The information is presented to the user in a single location, so it has to travel there at least once. However, centralized sampling may have some scalability problems, related to the network latency, overhead and resource usage.

After having the sampling values, sometimes it is necessary to perform some kind of processing to correlate them or to further enhance its meaning. This requires processor power and may also represent another scalability problem.

To cope with these problems, we suggest a distributed sampling mechanism based on mobile/static agents [8]. This should be seen as a dynamic tree of delegated sample processors. Their responsibility is threefold:

- Resolve syntax problems by defining a gateway between the management application and the agent system;
- Maintain a log of previous samples to provide historical information and to allow the remote processing of information;
- Perform basic processing operation on sample values.

The latter is seen as semantic compression because it extracts meaning of the raw data thus reducing the amount of information transmitted to the management

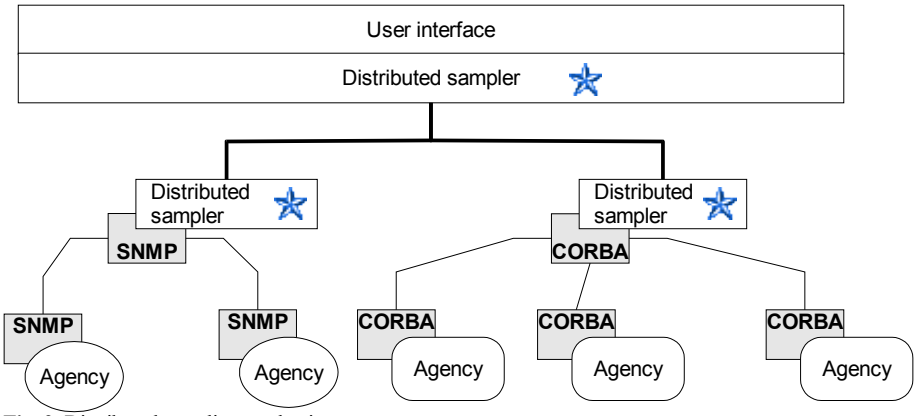

Fig. 2. Distributed sampling mechanism

console. To further cope with latency and to help saving network resources, the distributed samplers perform data compression by applying a lossless compression algorithm to the information before transmitting it (Fig. 2).

The hierarchy of distributed samplers defines several "islands" of independent sampling, thus reducing upper level network load and making possible the distribution of processing load among them.

The distributed sampler (DS) has the responsibility of sampling, logging and processing raw information retrieved from the agencies. The sampling operation requires it to retrieve values from the agencies by using a specific syntax and associates them with a time-stamp. The value is then logged to an internal database with user defined capacity based on time (space to store the values retrieved during two days operation) and on the number of samples (space to store one thousand samples). Finally, the information processing requires the DS to apply some kind of algorithm or mathematical expression to the logged values. One such operation is the evaluation of the difference between two consecutive samples: delta. Another feature

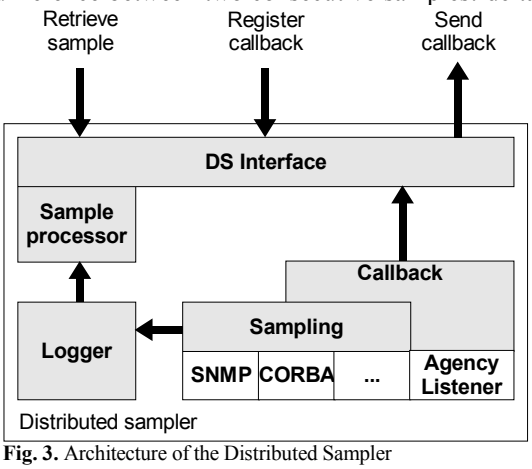

Fig. 3. Architecture of the Distributed Sampler

the DS should have is the possibility of registering callbacks to be notified when some value changes (Fig. 3).

In other words, the DS is a mobile agent with the capability of carrying different sample processors, a logger and different sampling mechanisms. This is easily achieved by defining attributes with the type of some base class and then initialized with an instance of the desired mechanism, such as SNMP or CORBA.

3 Graphical Elements

Graphical elements are used in the communication between applications and users. In the context of mobile agents the user may interact with the agents in two scenarios:

a) for introducing and retrieving information on mobile agents and

b) for managing agent systems.

In the first variant mobile agents interact with the user and, as such, they present a user interface for receiving user input. This situation introduces problems related to the agent platform nature and its capabilities as well as with the role of the user. If the agent is running on a regular PC, it can show up graphical windows; if it is running in a WAP terminal, it can show up WML (*Wireless Markup Language*) pages. The agent should also adapt the user interface according to the user role and permissions. For example, the system will present different features to a nurse and to a doctor. In other words, user interfaces to mobile agents should change the way they look according to were they are and to whom they are speaking to [9][10].

Agent system management, on the other hand, requires a specific application to monitor and control the operation of agents and agencies that will be based on the infrastructure information system. Almost every known MA implementation provides simple tools for controlling the agent life-cycle and for creating and destroying agents as well as places. However, the management of the supporting infrastructure, namely the agencies and hosts' resources, is usually marginally considered or not considered at all. Some related work focused on these problems at the API level but disregard the user interface level [11].

Each agency is characterized by a chart showing three parameters: the CPU load, the number of mobile agents and the number of static agents. The chart shows a set of values to build an idea of their evolution. When a new value arrives, the chart is updated by dislocating the values to the left (Fig. 4).

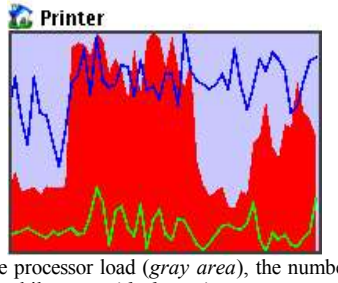

Fig. 4. Chart showing the processor load (*gray area*), the number of static agents (*light gray*) and the number of mobile agents (*dark gray*)

The main window has a search mechanism on the left which is used to locate the agencies registered in a specific region. The agency list below allows the user to position them by drag and drop on the map on the right. Then he can establish the sampling parameters for each one of them and start collecting values from the agency. A multi tabbed panel shows several layers which can represent a building floors, for example.

To get the full picture of the agents' and agencies' details, it is most useful to associate their position to a geographical map. The map may serve as background for the agencies and agents thus allowing the user to position them according to their location. A more complex scenario may include the tracing of mobile agencies together with a user location service to visualize, for example, the agency installed in his PDA or mobile phone (Fig. 5).

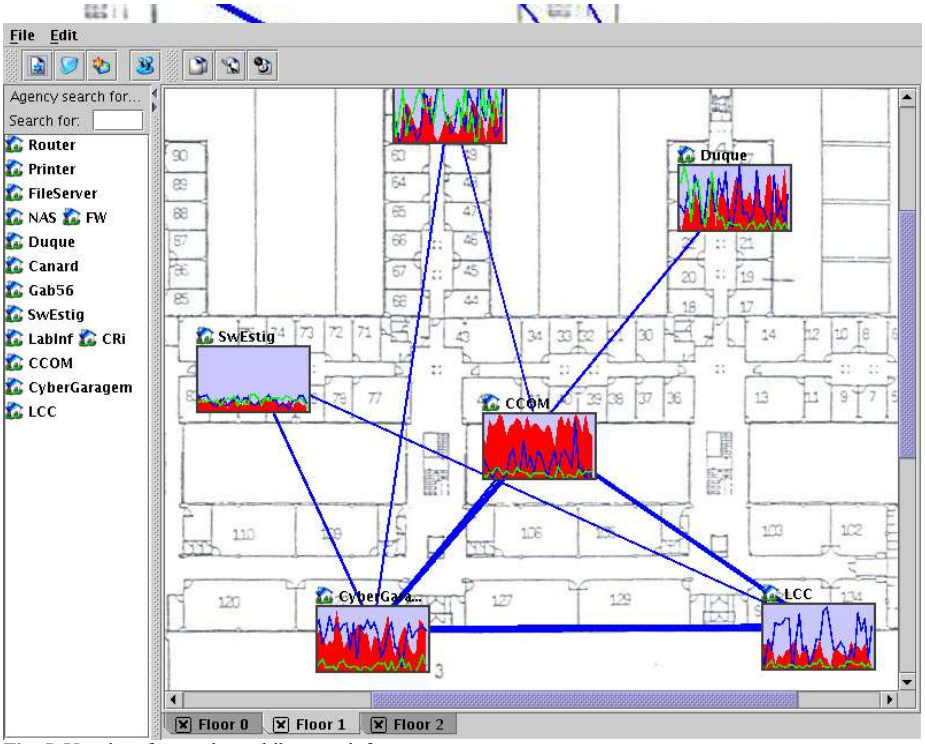

Fig. 5. User interface to the mobile agent infrastructure

In addition to these values, the interface also shows the connectivity between agencies. This parameter is measured as a rate of migrations between agencies, resulting from the evaluation of the number of migrations to a period of time. This value is then normalized as a percentage of all the migrations monitored and presented as lines. Different widths represent the relative percentage of migrations. The heavier the line, the heavier is the agent traffic between the connected agencies (Fig. 6).

To examine the details of agents and agencies, the user may call an explorer like tool [6]. The MAF Explorer follows the file manager paradigm with a tree view on the left and the content panel on the right. On the right side, it shows the details of the place or the agent, as selected on the tree. The grayed labels indicate that the parameter is read only. As an example, the user cannot modify the agent location, although the agent may move autonomously. A black label indicates that the user may also change its value. The agent status may be altered to suspend, resume or terminate its running status (Fig. 7).

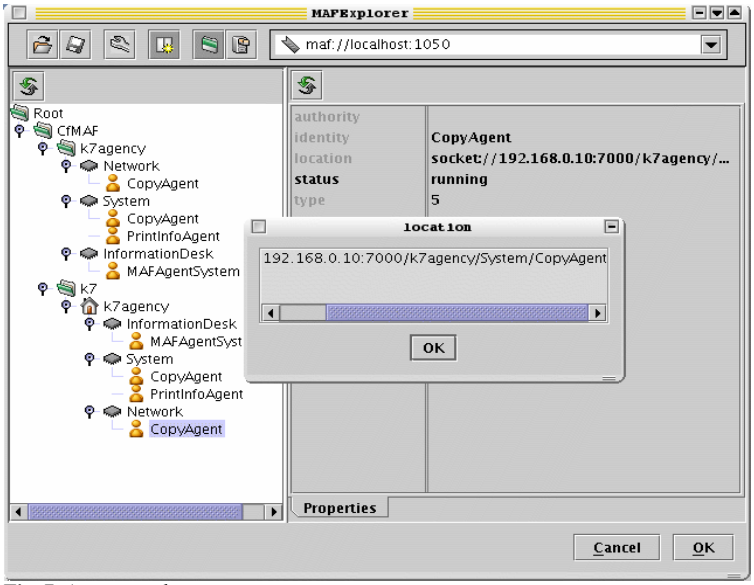

Fig. 7. Agency explorer

4 Implementation Details

The system is still being implemented at the time of writing of this paper, although some of the intended functionality is already working. Mobile agents support is provided by IKV's Grasshopper agent platform [12] and we are using the Sun's mapping of CORBA included in the J2SE [13] for the instrumentation objects.

The mobile agents information system is populated by the MAF interfaces and specific IDL interfaces to provide access to host instrumentation data. It was defined an IDL interface to define the mechanism to deal with host information and to implement the distributed callbacks mechanism. Data such as CPU load is retrieved from the agency host by a CORBA server object which also accept the registration of callback objects, so we have a scenario of mobile agents/CORBA to get all the information related to the navigation management system.

The monitoring system has to register a callback by acquiring a reference to the server object and then invoke the `register(callback)` method with a reference to the callback object. From this moment on it will be able to receive notifications.

5 Conclusions

In this paper we presented the approach, design and implementation issues of a mobile agent navigation manager. We discussed several related issues such as the mobile agent information system, the sampling strategies behind it and the set of graphical tools that build an intuitive graphical user interface.

Through the interface, the user will get access to the mobile agent system operation parameters and he will be able to start new tasks, modify existing ones and access information about how the system is working. We also presented concerns with the interoperability between the navigation manager with different vendors' mobile agent infrastructures by providing an all CORBA “driver” to the host and agency information.

References

1. Pham, V., Karmouch, A., "Mobile Agents: An Overview.", IEEE Communications Magazine, 1998.
2. Ibbotson, R., Gibbard, B., Stampf, D., Throwe, T., "A Mobile-Agent Based Performance-Monitoring System at RHIC", International Conference on Computing in High Energy and Nuclear Physics, Padova, Italy 2000.
3. Tierney, B., The Distributed Monitoring Framework (DMF), http://www-didc.lbl.gov/DMF/.
4. OMG, Mobile Agent Facility Specification, ftp://ftp.omg.org/pub/docs/formal/00-01-02.pdf.
5. Lopes, R., Oliveira, J., "SNMP Management of MASIF Platforms", proc. of the IFIP/IEEE International Symposium on Integrated Management - IM'2001, Seattle 2001.
6. Lopes, R., Oliveira, J., "Multi-management Schemes for MAF Platforms", proc. of the Fourth International Workshop on Mobile Agents for Telecommunication Applications (MATA'2002), 2002.
7. Simões, P., Silva, L., Boavida, F., "Integrating SNMP into a Mobile Agents Infrastructure", proc. of the Tenth IFIP/IEEE International Workshop on Distributed Systems: Operations & Management (DSOM'99), 1999.
8. Bohoris, C., Pavlou, G., Liotta, A., "A Hybrid Approach to Network Performance Monitoring Based on Mobile Agents and CORBA", proc. of the 4th International Workshop of Mobile Agents for Telecommunications Applications - MATA'2002, 2002.
9. Braubach, L., Pokahr, A., Moldt, D., Lamersdorf, W., "Using a Model-based Interface Construction Mechanism for Adaptable Agent User Interfaces", proc. of AAMAS Workshop 16 - Ubiquitous Agents on Embedded, Wearable, and Mobile Devices, 2002.
10. Silva, A., Silva, M., Romão, A., "User Interfaces with Java Mobile Agents: The AgentSpace Case Study", proc. of the First International Symposium on Agent Systems and Applications Third International Symposium on Mobile Agents, 1999.
11. Simões, P., Marques, P., Silva, L., Silva, J., Boavida, F., "Towards Manageable Mobile Agent Infrastructures", proc. of the International Conference on Networking - ICN'01, Colmar, France 2001.
12. IKV++, Grasshopper Mobile Agent platform, http://www.grasshopper.de/.
13. Sun, Java 2 Standard Edition, http://java.sun.com/j2se/.