Mobile Agents for Web-based Systems Management

Paolo Bellavista, Antonio Corradi, Fabio Tarantino
Dipartimento di Elettronica, Informatica e Sistemistica
Università di Bologna
Viale Risorgimento 2, 40136 Bologna, Italy
Ph.: +39-051-2093001; Fax: +39-051-2093073
{pbellavista, acorradi, ftarantino}@deis.unibo.it

Cesare Stefanelli
Dipartimento di Ingegneria
Università di Ferrara
Via Saragat 1, 44100 Ferrara, Italy
Ph.: +39-0532-293831; Fax: +39-0532-768602
cstefanelli@ing.unife.it

Keywords: Mobile Agents, Network and Systems Management, Security, Interoperability, Web-based accessibility.

Work carried out under the financial support of the Ministero dell'Università e della Ricerca Scientifica e Tecnologica (MURST) in the framework of the Project "MOSAICO, Design Methodologies and Tools of High Performance Systems for Distributed Applications".

Mobile Agents for Web-based Systems Management

Abstract

The increasing dimension and heterogeneity of global Web systems make harder their management with tools based on the client/server model. The Mobile Agent technology overcomes the limits of traditional approaches and proposes solutions that are suitable for the management of distributed and heterogeneous Internet-based systems. The paper describes the MAMAS environment and its implementation with a Mobile Agent technology. MAMAS has the goals of monitoring the whole system, introducing dynamic corrective actions and modifying system policies at run-time. MAMAS achieves these objectives by answering the guidelines of both security and compliance to standards. The choice of Java as the implementation language has permitted to achieve portability, to exploit the language security features, and to provide Web accessibility. The MAMAS compliance with CORBA ensures interoperability with legacy management platforms.

Keywords: Mobile Agents, Network and Systems Management, Security, Interoperability, Web-based accessibility.

Introduction

Many organisations face the problem of managing their distributed and heterogeneous systems that are composed by a large number of multi-vendor computing resources integrated in Intranet/Internet environments. This problem has leveraged the interest in general solutions for management: the goal is to efficiently handle the information about the whole system and to eventually control it. In this area, several organisations have produced a wide range of solutions. IETF and OSI have proposed a model of management based on the Client/Server (C/S) interaction (Case, 1990; Dickson, 1992). The interaction is usually statically decided, roles are a priori assigned and immutable, and the clients and the servers can only exchange data information. Decentralised

solutions can achieve a more scalable design of management systems, by extending the model with a hierarchy abstraction. In addition, the employment of Web-based interfaces has also improved the accessibility of several management tools and environments.

New execution models based on mobile entities (Fuggetta, 1998) have suggested novel approaches to network and systems management to face the increasing complexity of current distributed and heterogeneous systems (Yemini, 1996; Bieszczad, 1998). In particular, this paper describes a Mobile Agent (MA) approach to network and systems management. The proposed environment, called MAMAS (Mobile Agents for the Management of Applications and Systems), provides some distinguished properties: flexibility, dynamicity, Web accessibility, security, and interoperability (Bellavista, 1999).

The MAMAS management environment can adapt to very different organisation structures, in terms of either architectures or management policies. It provides a set of abstractions to model physical resources, because only different abstractions can suit the common localities of the Internet. The place, where agents can execute, represents the execution node. The domain represents the set of nodes in a common department. The gateway is the abstraction for interconnecting different domains to model a whole organisation composed by several departments.

In addition, MAMAS can suit different management structures to implement distributed and co-ordinated strategies, from the case of one central administrator in charge of the management of all resources, to the one of a group of administrators with distinguished responsibilities on different resources. Any administrator is associated with one or several roles (Lupu, 1997). Individual users may be dynamically associated and disassociated with roles, to enable rapid and flexible organisation changes, without altering the specification of the policies.

MAMAS administrators can configure and control the entire managed system from any node by exploiting Web-based Graphical User Interfaces (GUI) that are

available via any Web browser. MAMAS also permits to introduce automatic responses to management problems and to dynamically inject new behaviour into the system, either to solve unforeseen situations or to adapt to new requirements.

MAMAS considers security a crucial aspect of the design, and integrates it at any system layer. Only this pervasive approach can achieve a quality level different from the minimal one obtained by systems that add a security strategy a posteriori. Our management environment makes available a wide range of security mechanisms and tools, to grant security to both agents and execution nodes. Administrators can choose the most suitable security level, taking into consideration the cost of the selected tools in terms of performance.

In addition, interoperability has guided the MAMAS project in recognising the importance of the Internet standards and in offering services integrated with them. This guideline has led to take into account TCP/IP services, and to implement CORBA compliance (OMG, 1995). We have selected the MASIF proposal (GMD, 1998) to enhance MAMAS interoperability with CORBA-based management tools (TIVOLI, 1997).

MAMAS is based on the MA model: mobile agents act on behalf of administrators and can move in the network to operate locally to resources and manage the distributed system (MAMAS is available at `http://www-lia.deis.unibo.it/Software/MA/`). From the implementation point of view, MAMAS has inherited from the support, realised in the Java language (Gosling, 1996), the properties of portability, interoperability, rapid prototyping, and easy integration with the Web scenario.

The paper gives an overview of the solutions in the field of network and systems management and presents our MA-based approach to management. Finally, we present and evaluate the performance of the MAMAS mechanisms.

A Comparison of Systems Management Approaches

The OSI committees are still engaged in the definition of standards in the management area. The handling of large and heterogeneous distributed systems is complex, and it still raises discussions at either the committee level or the industrial one, e.g., OSI (Dickson, 1992), TMN (Glitho, 1995), TINA (Inoue, 1998), IETF (Case, 1990).

Standard Management Approaches

The most common solution to the management problem derives from the Internet widespread protocol, SNMP (Case, 1990), a very basic protocol to exchange management information. Several SNMP-based products give a central view of the state of a distributed system: OpenView (HP, 1992), NetView (IBM, 1997), NetManager (SUNSOFT, 1994) give the operator complete information about the managed LANs and their interconnections. Only a limited possibility of automated actions is available: most of the installations require the presence of an operator to take real-time decisions. In addition, they are closed tools, in the sense that changes in the organisation policies can be difficult to accommodate. More recent products (TIVOLI, 1997) start addressing these issues and define the interaction according to new standards, such as CORBA (OMG, 1995).

In general, the tools above are good examples of design, but they are based upon the C/S model, intrinsic to SNMP: a central manager controls several remote agents. The central manager provides a proprietary user interface to the system administrator and interacts with the entities that run on remote nodes to manage the local management information. The interaction uses a fine-grained C/S management protocol, and is likely to introduce overhead, the so-called micro-management problem: a high volume of traffic is generated around the central manager node, also overloaded in controlling the whole computation.

While SNMP agents are very simple computational entities that reside one for each network node, our mobile agents can become powerful computation entities that can move autonomously from node to node, acting on behalf of the system

administrator. MAMAS agents can not only collect the network node information in order to show the overall situation to the system administrator but also perform complex administration tasks on the managed nodes. Agents can work to back-up file systems, shutdown either predefined or dynamically determined nodes, install new network services, etc. The decentralisation intrinsic to the MA model can avoid the central manager bottleneck of traditional approaches.

Management Approaches Based on UNIX and Script Languages

Several Unix-based management tools start from scratch to provide management features, instead of using standard protocols. The goal is to give to a central operator a general view of the current system state, with a limited possibility of automatic and manual intervention (Finkel, 1997). These projects define a scenario for rapid development, by using implementation languages such as Perl (Wall, 1990) and Tcl/Tk (Welch, 1997). While the advantages of shell languages are mainly concentrated on the possibility of rapid application development, their usage makes difficult to integrate all the realised features in a unique environment able to solve the management problems of one organisation.

Novel Management Approaches

In the last few years, several researches have focused on the possibility for distributed systems to host mobile and dynamic entities. Different answers to systems management come from these new models of execution (Fuggetta, 1998; Stamos, 1990; Baldi, 1997; Leppinen, 1997).

Management by Delegation (MbD) represents a clean effort toward decentralisation and increased flexibility of management functionality (Yemini, 1996; Goldszmidt, 1995). MbD dynamically distributes network management components to extensible remote managers that can learn new modes to handle resources. MbD can also integrate with existing management protocols, like SNMP. While the MbD is shaped after the Remote Evaluation programming

model, the MA model has broader capacity in describing mobility and subsumes MbD (Fuggetta, 1998).

Another innovative approach to network management has the goal of obtaining plug-and-play networks (Bieszczad, 1998). This research proposes the use of mobile code for dynamic network configuration and presents several applications of mobile code. The infrastructure includes Java-based mechanisms for code mobility, security management and communication. There are many similarities between that framework and MAMAS. We stress the mobile agent approach, by providing a general MA-based support with a proper security model suited to many interconnected environments and with the goal of interoperability, as described in the following sections.

The MAMAS Environment for Management

Different organisations have different management policies and have to deal with different distributed system architectures. Management policies derive from the organisational attitude about security, service availability, fault tolerance, quality of service, etc. The available configurations vary from systems of one simple LAN to systems composed of several LANs variously interconnected by bridges, routers, gateways, and firewalls.

One organisation may consist of several departments, even geographically distributed over the Internet. Each department with its own LANs and resources should interact via gateways with other departments. When departments have to communicate via the Internet, the same levels of security and QoS as in Intranet communication should be granted. From an administration perspective, while simple traditional approaches tend to identify a central administrator in charge of managing all resources, many organisations are better suited to distributed and co-ordinated strategies, with several administrators in charge of different responsibilities on different resources.

The MAMAS environment exploits the MA technology to realise a flexible solution to the systems management problems. MAMAS originally contributes to the management area because of its intrinsic capacity of moving execution entities to the part of the system to be controlled, thus overcoming the restriction of the traditional C/S model of interaction.

In addition, MAMAS is a tool designed for open systems, and follows the guideline of compliance with the Internet and Web standards. It is also able to request services from CORBA compliant management tools (OMG, 1995), and we are currently extending its implementation to achieve full interoperability with legacy management systems.

The MAMAS Organisation

MAMAS offers an easy way to answer different management issues by introducing two orthogonal dimensions in configuration (see Figure 1):

- network locality, i.e., the set of abstractions to model the physical resources;
- administration locality, i.e., the responsibility domain of one system administrator.

MAMAS models *network localities* by defining several abstractions for physical resources. The place abstraction, where agents can execute, represents the physical machine. The domain abstraction encloses a set of places; it typically represents a LAN and includes a gateway abstraction, called default place, that is in charge of any possible interconnection between different domains.

MAMAS models any kind of *administration locality*, by grouping the resources controlled by each administrator. Several administration localities may overlap, thus modelling the joint management of some resources by several administrators, even with different permissions over the same resource.

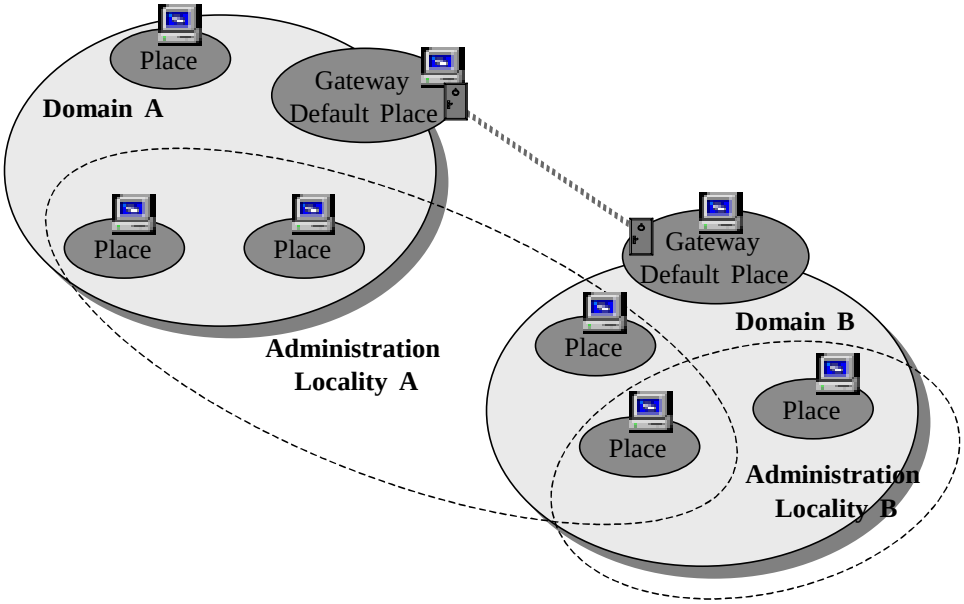

Figure 1. MAMAS abstractions for network and administration localities.

The MAMAS ability of modelling both network and administration localities offers a powerful way for implementing several management policies. At one extreme, it is possible to configure MAMAS to adapt to a simple centralised management scheme where a single administrator controls one single LAN network locality. At the other extreme, it is possible to configure it for one organisation where different system administrators have different administration duties; the managed system can be composed by a multiplicity of network localities, e.g., domains interconnected by gateways (and firewalls) and connected to the Internet. The MAMAS environment helps in the management of all the above situations and any other intermediate, by taking into account the related security requirements, as described in the following.

In MAMAS, each administration authority represents a role with specific authorisations for the access to different resources: the use of roles is justified to enable rapid and flexible organisational changes. In fact, administrators may be associated and disassociated with administration roles without altering the specification of the policies (Lupu, 1997). Any administrator can control the whole system by creating agents that move within her administration locality.

Managed resources are capable of accepting/refusing operations to agents depending on the agent administration role. Any resource has its specific access control list for all roles. Let us consider two different administration roles, of different level of responsibility: the organisation-manager and the department-manager. The former can send management agents to all hosts in the organisation, while the scope of the latter is limited to a specific department *A*. They share the possibility of operating on the *A* system resources. For instance, a possible shutdown action that switches off some executing resources can be commanded from any of the two authorities. The shutdown could also be graceful, to grant a minimal level of operations and to maintain a few services available in the target department *A*, for instance, one HTTP, one FTP, and two database servers. In this example, the minimal set of services results by merging the two requirements, of both the organisation-agent and the department-agent. In case of conflicting requirements, the organisation-manager role prevails on the lower priority department-manager one.

The MAMAS Agents

Our MA approach potentially avoids any centralisation point and provides better fault tolerance and scalability than centralised C/S scheme. Several administrators can be concurrently active and even co-operate to obtain a single administration goal. It is easy to generate/destroy agents and to replicate them in case of a large node number in the locality.

In MAMAS, agents act on behalf of administrators and fulfil administration needs by moving and executing on different nodes. Any administrator can implement her policy by using agents. The MAMAS environment provides a rich set of already defined agents for systems management. In addition, it is easy to tailor new agents to new specific administration needs in order to delegate the automation of new management tasks. The following list gives a few examples of already implemented functionality of agents in the MAMAS environment. MAMAS agents can:

- monitor the state of the distributed system;
- help in the configuration of any new or reinserted node;
- easily be used in the control and co-ordination of replicated resources;
- be in charge of the shutdown of the whole system and can also guarantee a minimal survival service level;
- regulate and improve the access to different databases by taking into account both the traffic level and the locality of the queries;
- dynamically install new communication protocols for new applications.

As an example of a MAMAS agent, let us consider the monitor agent that can report to one administrator the information about the state of the whole system. The agent gives the situation of each node in terms of system and application indicators (see Table 1). In addition, it can report network management information, such as the measured collision rate.

System indicators		Application indicators
CPU load	collision rate	availability of services
file system occupation	network connectivity	program versioning
swap space available	firewall state	application processes situation
daemon process situation	...	local agent states
printers status	...	...

Table 1. Some indicators available in MAMAS.

MAMAS makes possible to delegate specific controlling actions to agents, thus relieving the administrator duty. For instance, one agent can automatically take care of software upgrading on all the nodes of one domain. Another distinguished feature of MAMAS is the capacity of modifying system policies at run-time. When a policy modification interests several nodes, there is no need to shutdown the whole system: a new agent can bring the new policy everywhere. The same run-time propagation applies to any static function.

MAMAS Web-based GUI

Any administrator can access to the MAMAS environment via any Web browser. In fact, MAMAS provides a user-friendly graphical interface to operate directly on the system. For example, Figure 2 shows how the administrator can control the initial configuration of the system and its modification at run-time. Any administrator is first authenticated, and then authorised to perform different operations depending on her role. The same interface permits administrators to handle new roles and administrators, to add new places and domains to the system, and to provide new resources and behaviour.

Figure 3 describes the GUI offered by the monitoring tool to report the state of a specific host. The administrator is given the situation of a node in terms of system and application factors, e.g., the situation of physical resources, such as CPU and disk occupation. The application monitoring part permits to create and send new agents where requested in the distributed system.

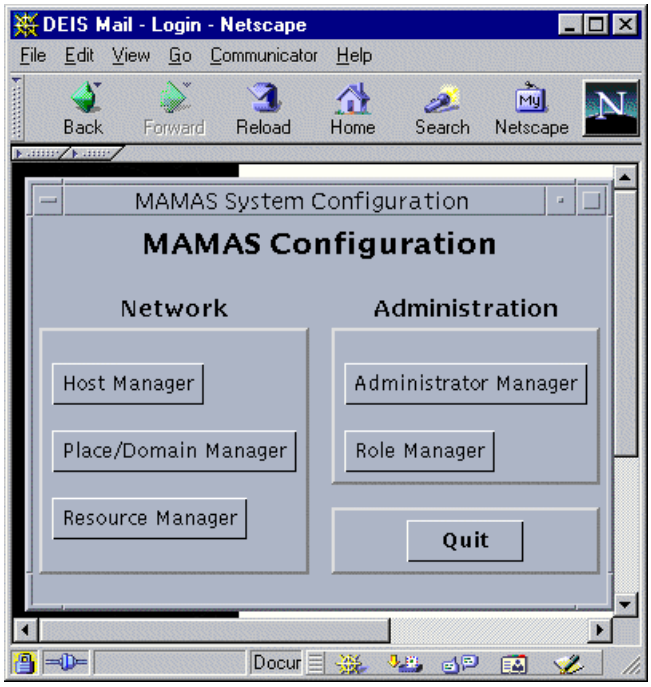

Figure 2. The MAMAS Web-based GUI for locality configuration.

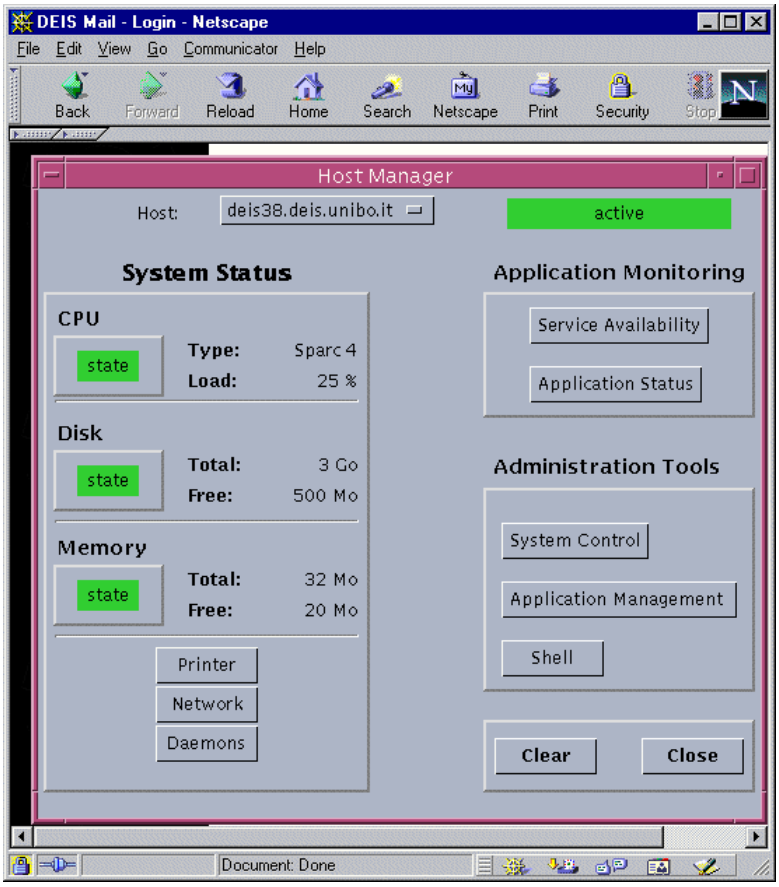

Figure 3. The MAMAS Web-based GUI for distributed monitoring.

Security in MAMAS

The typical MAMAS managed system consists of Internet-based nodes and should face the security problems induced by this untrusted environment. In addition, agents involved in systems management perform very sensible operations. They have critical system duties and the risk of an incorrect action due to unwanted and malicious reasons is intolerable. To prevent any malicious action, the MAMAS security framework protects places, by verifying the integrity of the incoming agents, by authenticating and associating them with their recognised administration role, and by controlling agent operations on resources.

To protect resources, agents can perform local operations only by requesting services to object interfaces because direct access to resources is ruled out. On the one hand, this separation respects the encapsulation principle, and on the other hand, it achieves agent independence from resource implementation,.

In addition to the protection of places and resources, agent protection forces to consider two different scenarios, with very different security threats and requirements: Internet-based environments and Intranet ones.

In the case of Internet environments, agents could be attacked when they migrate and traverse insecure paths. There is no prevention against the destruction of an agent that migrates through possibly hostile channels. MAMAS overcomes this problem via message numbering and confirmed sending operations, with a periodical use of a checkpointing technique to save the agent state (Peine, 1997) and to recover from the previous state in case of a loss. To ensure secrecy, management agents can be encrypted when traversing an insecure path with secret keys known to the sender/destination places. The detection of any agent malicious modification occurs via the use of a secure hash function verified in the integrity check phase (see Figure 4). When one agent enters a place, the security infrastructure identifies the principal responsible for the agent by means of its signatures and associates the agent with its responsible administration role (authentication and authorisation checks).

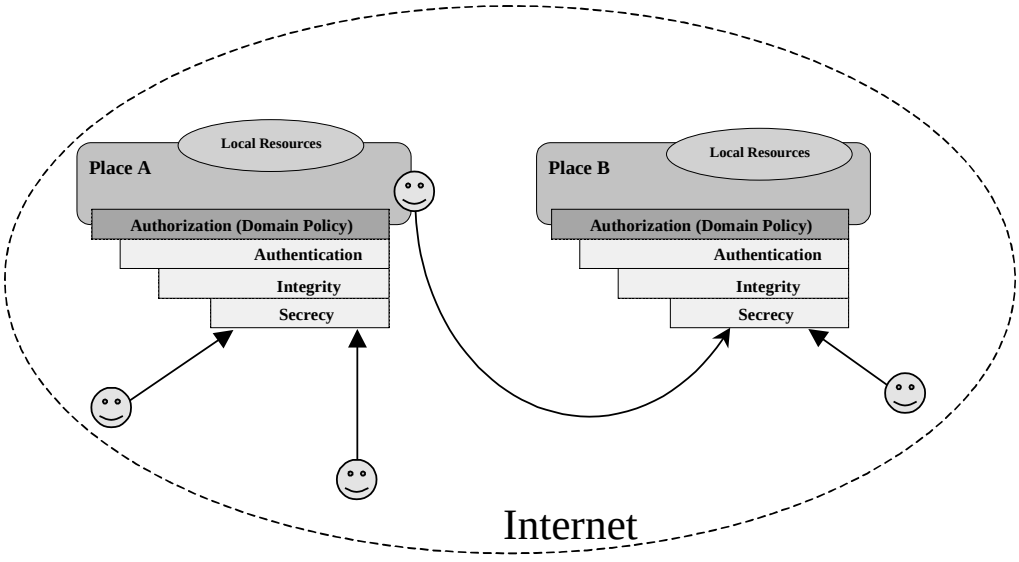

Figure 4. Security controls for agents while entering a place.

Intranet environments are intrinsically more protected. In this scenario, some security checks can be avoided, and their cost saved. For instance, in a protected

environment the secrecy check can become unnecessary. In case of reliable environments that grant a dependable message delivery, the integrity check can be superfluous. In case of co-operative environments, where there is a complete trust among all involved administration authorities, even signatures can be saved. In that way, MAMAS can provide the security level suitable to any Internet and Intranet need.

The Implementation of MAMAS Agents

Agents can move from node to node and access resources and services of the currently local place by commanding place objects. Figure 5 shows the code of a simple agent for monitoring the CPU load of the network nodes.

```
void run(){      // starting method for every Agent

     // Asking to AgentSystem the list of Nodes in this Domain
     Node = AgentSystem.getAllDomain();
     CurrNode=0;
     // Looking for the first active Node
     for(;CurrNode<Node.length;CurrNode++)
          if(AgentSystem.isActive(Node.Name)) goNode();
     ... // Error: no active nodes
}

void goNode(){
     try{
          This.go(Node[CurrNode].Name,"VerifyNode");
     } catch(Exception e){//Can't go, System or Security exception
          ...
          goHome(); // Back home with failure status
     }
}

void VerifyNode(){ // Restart method specified by goNode()
     try{
          CPUload[CurrNode]=Monitor.getCPULoad();
     } catch(Exception e) { // action not allowed
          ... // Actions for exception handling
     }
     CurrNode++;
     for(;CurrNode<Node.length;CurrNode++)
          if(AgentSystem.isActive(Node.Name)) goNode();

     goHome();
}
```

Figure 5. A simple MAMAS agent to monitor the CPU load of a set of specified nodes.

In this example, the agent visits all nodes and ascertains the load of the current one by calling a specific method of the `Monitor` place object. While the implementation of the place objects is system-dependent, the agent code is independent of hardware/software architecture.

An additional example is the case of one agent that collects the information about the file system occupation. It can request the `diskusage` method of the `Monitor` object, that calls in its turn the system dependent command, e.g., a `df` Unix command on a Sun workstation and a `bdf` in an HP one.

The Mobile Agent Support

Several MA environments are available from different sources (Huhus, 1997; Rothermel, 1997; Vitek, 1997). Their agents can move from node to node and can co-ordinate via either message passing or shared resources inside a node.

We have developed an MA support from scratch, in order to fully support some fundamental properties of MAMAS: flexibility, security and interoperability. The adopted MA support provides a hierarchy of locality abstractions suitable for describing any kind of internet-worked scenario (Bellavista, 1999). Any node has a place for agent execution; several places are grouped in domain abstractions that can be interconnected by using gateways. The MA support provides also a range of security mechanisms: authentication of mobile agents on the basis of public key certificates, integrity check on the code, secrecy of agent status, and controlled resource access based on authorisation, ACL mechanisms and safe interfaces. The properties of portability and interoperability are ensured by the use of Java as the implementation language and by the choice of CORBA compliance (OMG, 1995).

With regard to agent co-ordination, agents inside a place can interact by sharing common resources. Whenever one agent needs to share one resource with another agent residing in a remote place, it is forced to migrate to that remote place. Outside the scope of the place, agents can interact only via message exchange. Messages are eventually delivered to agents even in case of migration. Any other

advanced scheme of communication and co-operation can be implemented on top of these basic mechanisms. The MA support provides the name system to ensure the message delivery to agents: it is based on a federation of name servers, one per domain, each in charge of answering the requests generated in its locality. We have initially adopted an *ad-hoc* naming solution, with the goal of integrating with accepted naming standards, such as DNS and CORBA Naming.

Agents are transferred from place to place (possibly in different domains) by using a simple communication protocol to serialise/deserialise agents, while waiting for an agreement for the definition of a standard MA exchange protocol (GMD, 1998; Lange, 1997).

Our MA support uses the JDK 1.2 (Sun, 1999). We have developed it on SUN workstations, and easily ported it to PCs. The OO nature of the Java language has helped in the design of the MA platform. The encapsulation principle suits the abstraction needs of both resources and agents; the classification principle makes possible to inherit behaviour from already specified components; multithreading, garbage collection and error management simplify writing robust code.

A well-known problem of Java is the lack of full mobility support, especially for Java threads: it is not possible to save the whole state of a thread before its migration to a different node. This restriction can be overcome by either modifying the Java Virtual Machine or providing a new operation at the application level. We have chosen the latter solution to preserve portability. The new operation for mobility is a `go` operation that allows one agent to move itself during its execution, by specifying the method to be activated after the migration.

From the point of view of security, the MA support provides a variety of mechanisms to grant security to the MAMAS entities. The agent authentication exploits the DSA algorithm and X.509 certificates. Agent integrity and secrecy are achieved by standard cryptographic mechanisms: the iSaSiLk package provides both DES channel encryption and the SSL protocol solution (IAIK, 1998).

MAMAS Performance

MA management solutions can obviate to the micro-management problem and can delegate management activities to agents. The MAMAS implementation has shown that MA solutions can also be convenient from the point of view of performance.

In traditional solutions, the central manager resides on one node and collects the monitoring information of the whole domain by message exchange with remote servers, one in each controlled node (see Figure 6). MAMAS favours decentralisation: it achieves better results than the centralised approach as soon as the operations performed by each agent counterbalance migration costs.

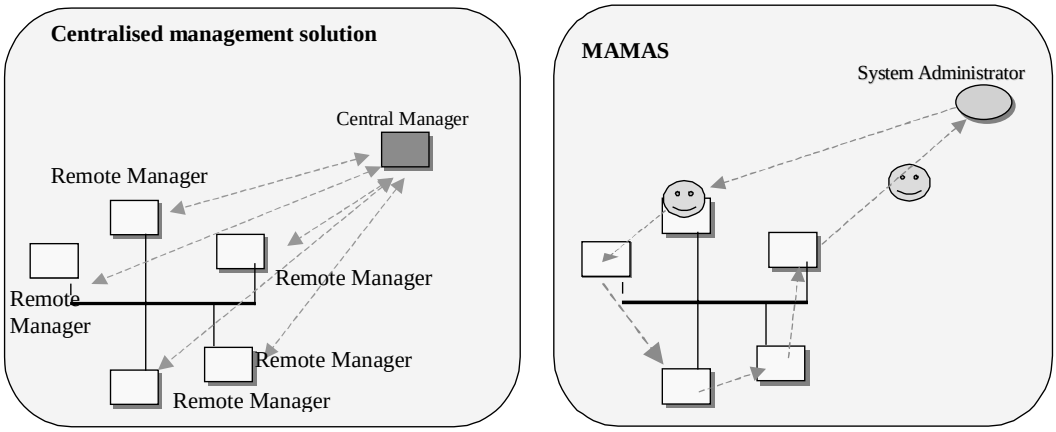

Figure 6. Different approaches to systems management: the traditional centralised solution vs. the decentralised MA-based one.

Let us consider the example of a central manager that needs to ascertain the consistency of the software versions of all administrated places. We have measured the cost to inspect the configuration files for all places. The traditional C/S solution requires a number of messages equal to the number of files to be controlled (for each remote server). In the agent solution, the system administrator dispatches a variable number of monitoring agents to control the situation in the domain; each agent verifies the consistency locally and reports only the final situation to the operator. Table 2 shows the total time (in msec) to complete the task in a domain composed of 12 Ethernet-connected workstations (SUN Sparc 5,

Ultra-1 workstations and PCs). The experiment has been performed within an Intranet that is protected from the Internet by a firewall. The Intranet is an example of a co-operative and trusted domain: agents need neither signature nor encryption while migrating. The measured time is an average over a high number of executions.

The MA solution can perform better than traditional ones as soon as the manager employs a suitable number of monitoring agents in the administrated domain. The saturation effect in Table 2 stems from the cost of agent creation not balanced by effective agent operations. Apart from performances, the MA-based approach can offer a solution more suitable than traditional ones for dealing with mobile computing, network bandwidth limitations and unreliable communication scenarios.

No. of monitoring agents	Time in MA solution (msec)	Time in C/S solution (msec)
1	215	188
3	129	
6	98	
9	112	
12	131	

Table 2. The MA-based monitoring solution compared vs. traditional C/S monitoring.

When the agent migration needs Java classes that are not present at the agent destination, it is necessary to provide mechanisms to load these classes. The loading operation, which occurs at any time in case of applet execution, is necessary only once in MAMAS, and its impact on performance is limited if the same behaviour is used several times. In addition, the cost for agent migration is strongly influenced by the costs of the required security mechanisms. We have extensively tested the loading and security mechanisms of MAMAS to give an idea of the cost of moving agents under different assumptions:

- class loading before operating at the destination node (agent code migration);
- class loading at the same time of agent loading (agent + class code migration);
- agent migration over secure/insecure path.

Table 3 and Table 4 report the migration costs for different-sized agents, in different platforms, in the case of secure/insecure path. In both tables, the first row shows the cost of code migration when the class code is already present in the destination node. The second row gives the migration cost when the agent carries also its classes with it, and the third one the cost for an even larger agent: the larger the bytecode to move, the higher the cost. When migrating on secure paths (i.e., in trusted domains), the agent is transferred between places with no encryption/decryption overhead. In the case of an insecure path, MAMAS encrypts agents with DES to ensure agent secrecy. The reported results indicate clearly the high costs of encryption: this overhead stems also from the use of the Cryptix package (version 2.2), that operates partly in native and partly in interpreted code.

Table 5 indicates the cost associated with signature verification that is necessary to ascertain the agent principal and authorisation: let us note that is almost independent of agent size.

Agent Dimension (byte)	Secure Path (msec)	Insecure Path (msec)
218 (agent code)	21	180
1438 (agent code + class)	61	710
7300 (agent code + class)	176	4520

Table 3. Migration costs between Ethernet-connected SUN SPARC 5 workstations with Solaris 2.5.

Agent Dimension (byte)	Secure Path (msec)	Insecure Path (msec)
218 (agent code)	17	147
1438 (agent code + class)	40	461
7300 (agent code + class)	120	2924

Table 4. Migration costs between Ethernet-connected PC Pentium 133 MHz with Windows NT 4.0.

Agent Dimension (byte)	NO Credential (msec)	One Credential (msec)
1438 (agent code + class)	61	438
7300 (agent code + class)	176	552

Table 5. Migration costs in case of signature verification between Ethernet-connected SUN SPARC 5 workstations with Solaris 2.5.

Conclusions

The paper presents MAMAS, a systems management environment based on the MA paradigm. Apart from monitoring the distributed state of the system and visualising it to the operator, it favours the automation of several management actions and permits to dynamically change the predefined system policies. These services are achieved by answering important requirements such as hardware and software heterogeneity, flexibility, rapid development, and efficiency.

The MAMAS tool stresses two project guidelines, security and interoperability, to achieve acceptance. We have decided to answer the security requirement at any level of the project and with different security degrees that permit to administrators to take into account also the resulting service performance. We have decided to strive for interoperability with recognised standards to grant the expected durability of the design effort. At the level of accessibility, MAMAS suggest the usage of Web-based tools. At the level of user services and command, any management function can be embodied by suitable agents. At the level of openness, the CORBA interface permits to interoperate with CORBA-based management tools. In addition, the use of mobile agents makes possible to dynamically modify the behaviour of several MAMAS components and to dynamically check different policies.

The Java implementation, apart from the a priori granted portability, has also re-established the possibility of rapid prototyping. The choice of Java has leveraged the integration with Web-based management components and Internet tools.

References

Baldi, M., Gai, S., and Picco, G. (1997), "Exploiting Code Mobility in Decentralized and Flexible Management", *Proc. 1st International Workshop on Mobile Agents*, Berlin (D), Lecture Notes in Computer Science, No. 1219, Springer-Verlag.

Bellavista, P., Corradi, A., Stefanelli, C. (1999), "An Open Secure Mobile Agent Framework for Systems Management", to be published in *Journal of Network and Systems Management*, Vol. 7, No. 3.

Bieszczad, A., Pagurek, B., and Susilo, G. (1998), "Infrastructure for Advanced Network Management based on Mobile Code", *IEEE/IFIP Network Operations and Management Symposium NOMS'98*, New Orleans, Louisiana.

Case, J., et al. (1990), Simple Network Management Protocol (RFC 1157), DDN Network Information Center, SRI International.

Chiariglione, L. (1997), *FIPA 97 Specification*, Foundation for Intelligent Physical Agents, http://www.fipa.com/.

Dickson, G., and Loyd, A. (1992), *Open Systems Interconnection*, Trevor Housley (Ed.), Prentice-Hall.

Finkel, R.A. (1997), "Pulsar: an Extensible Tool for Monitoring Large Unix Sites", *Software Practice and Experience*, Vol. 27, No. 10.

Fuggetta, A., Picco, G.P., and Vigna, G. (1998), "Understanding Code Mobility", *IEEE Transactions on Software Engineering*, Vol. 24, No.5.

Glitho, R. H., and Hayes, S. (1995), Special Issue on "Telecommunications Management Network", *IEEE Communications*, Vol. 33, No. 3.

GMD FOKUS, IBM Corp. (1998), *Mobile Agent Facility Specification*, Joint Submission supported by Crystaliz Inc., General Magic Inc., the Open Group, OMG TC Document orbos/97-10-05, ftp://ftp.omg.org/docs/ orbos/97-10-05.pdf.

Goldszmidt, G., and Yemini, Y. (1995), "Distributed Management by Delegation", *Proc. 15th International Conference on Distributed Computing Systems*, IEEE Computer Society, Vancouver, British Columbia.

Gosling, J., Joy, B., and Steele, G. (1996), *The Java Language Specification*, Addison-Wesley, Manlo Park, CA.

Hewlett Packard (1992), *Openview User's Guide*, HP.

Huhns, M.N., and Singh, M.P. (1997), Special Issue on "Internet-Based Agents: Applications and Infrastructure", *IEEE Internet Computing*, Vol. 1, No. 4.

IAIK (1998), *iSaSiLk 2.0*, http://jcewww.iaik.tu-graz.ac.at/iSaSiLk/.

IBM (1997), *NETView*, http://www.networking.ibm.com/netprod.html.

Inoue, Y., Guha, D., and Berndt, H. (1998), "The TINA Consortium", *IEEE Communications*, Vol. 36, No. 10.

Lange, D.B., and Aridor, Y. (1997), *Agent Tranfer Protocol–ATP/0.1*, IBM Tokyo Research Labs, http://www.trl.ibm.co.jp/aglets/atp/atp.html.

Leppinen, M., et al. (1997), "Java- and CORBA-based Network Management", *IEEE Computer*, Vol. 30, No. 6, pp. 83-87.

Lupu, E., and Sloman, M. (1997), "A Policy Based Role Object Model", *Proc. EDOC'97*, IEEE Computer Society Press.

Object Management Group - OMG (1995), *The Common Object Request Broker: Architecture and Specification*, Rev 2.0, OMG Document 96-03-04.

Peine, H. (1997), "An Introduction to Mobile Agent Programming and the Ara System", *ZRI-Report 1/97*, Dept. Of Computer Science, University of Kaiserslautern, Germany.

Rothermel, K., and Popescu-Zeletin, R. (1997), *Proc. 1st International Workshop on Mobile Agents*, Berlin (D), Lecture Notes in Computer Science, No. 1219, Springer-Verlag.

Stamos, J.W., and Gifford, D.K. (1990), "Remote Evaluation", *ACM Transaction on Programming Languages and Systems*, Vol. 12, No. 4.

SUN Microsystems (1999), *Java Development Kit - Version 1.2*, http://java.sun.com/products/.

SUNSOFT (1994), *Sun NetManager*, Reference Manual, SunSoft Press.

Systemics (1997), *Cryptix*, http://www.systemics.com/software/cryptix-java/.

TIVOLI (1997), http://www.tivoli.com/.
Vitek, J., and Tschudin, C. (1997), *Mobile Object Systems Towards the Programmable Internet*, Lecture Notes in Computer Science, No. 1222, Springer-Verlag.
Wall, L., and Schwartz, R. (1990), *Programming Perl*, O'Reilly.
Welch, B. (1997), *Practical Programming in Tcl and Tk*, Prentice Hall.
Yemini, Y., and da Silva, S. (1996), "Towards Programmable Networks", *IFIP/IEEE International Workshop on Distributed Systems: Operations and Management*, L'Aquila, Italy.

Paolo Bellavista received his Laurea in electronic engineering from the University of Bologna, Italy, in 1997. He is currently pursuing a Ph.D. in computer science engineering at the same university. His research interests include distributed computing, distributed objects, mobile agents, network and systems management, multimedia systems for distance learning. He is member of the ACM and IEEE.

Antonio Corradi is an associate professor of computer science at the University of Bologna. His scientific interests include distributed systems, object and agent systems, network management, and distributed and parallel architectures. He received his Laurea in electronic engineering from the University of Bologna and his MS in electrical engineering from Cornell University. He is member of the ACM, AICA (Italian Association for Computing), and IEEE.

Cesare Stefanelli received his Laurea in electronic engineering from the University of Bologna, Italy, in 1992 and the Ph.D. degree in computer science in 1996. His research interests are in the area of distributed systems, massively parallel systems and programming environments for parallelism. Currently, he is an associate professor of operating systems at the University of Ferrara. He is member of the AICA (Italian Association for Computing), and IEEE.

Fabio Tarantino received his Laurea in computer science engineering from the University of Bologna, Italy, in 1998. His research interests are in the area of distributed systems, object-oriented programming, network and systems management. He currently works for Andersen Consulting.