ELSEVIER

Microprocessors and Microsystems 25 (2001) 75–83

MICROPROCESSORS AND MICROSYSTEMS

www.elsevier.nl/locate/micpro

Middleware services for interoperability in open mobile agent systems

Paolo Bellavista[a,*], Antonio Corradi[a], Cesare Stefanelli[b]

[a]*Dip. Elettronica, Informatica Sistemistica, Università di Bologna, 40136 Bologna, Italy*
[b]*Dip. Ingegneria, Università di Ferrara, 44100 Ferrara, Italy*

Received 30 January 2001; accepted 1 February 2001

Abstract

Despite the design and implementation of several mobile agent (MA) platforms, widely diffused services based on the MA programming paradigm are still lacking. Apart from the security challenges imposed by the MA technology, the paper claims that interoperability between MAs, legacy systems and heterogeneous MA platforms is a major obstacle to the MA diffusion. The paper discusses solutions to permit the interworking between MA platforms and other systems, even non-MA-based, via compliance with either accepted or emerging interoperability standards. In particular, it focuses on compliance with CORBA, the accepted standard for object-oriented components, but also with MASIF and FIPA, respectively, the OMG specification for the support of agent mobility and management, and the framework for standard agent platforms and communication languages. The discussed solutions have guided the design and implementation of the middleware interoperability service in the secure and open mobile agents (SOMA) programming framework. The SOMA interoperability service is structured in a layered and modular way: its components can be dynamically distributed and installed only where and when needed. The paper also presents an application scenario in the area of network, systems and service management, where the interoperability components permit the interworking of SOMA agents, SNMP-compliant elements, legacy resources, and non-SOMA MA platforms.

Keywords: Mobile agents; Standards; Interoperability; Middleware; CORBA

1. Introduction

The widespread popularity of the web and the ubiquitous availability of internet access points stimulate the idea of global network as an open distributed system for the design, implementation and deployment of services. These services are the result of the dynamic composition and interworking of networked components, and force us to consider a scenario where distribution, dynamic modifications, heterogeneity and openness are basic requirements.

The provision of globally available web services has stimulated the research on several emerging programming paradigms. Some of them (remote evaluation, code on demand and mobile agents (MAs) [1–4]) move from the consideration that the traditional client/server (C/S) model is not flexible enough for the new scenario. They extend the expressive capacity of the C/S paradigm by giving the possibility of transferring the code over the network at run-time. In particular, the MA paradigm permits location-aware executing entities to migrate from one network host to another while in execution, by carrying with them both code and required execution state [1].

MAs are supposed to provide attractive solutions in many application areas, as presented extensively in Ref. [5]. Till date, however, while there are several different MA systems, there are still only a few MA-based commercial services, and most of them are limited case studies. What is currently slowing the MA diffusion in commercial applications is the impression of immaturity still around the MA technology. On the one hand, the large number of different MA systems induces a sense of uncertainty. System diversity is important but can produce incompatibility and discourage related investments. On the other hand, the MA paradigm certainly imposes to solve specific and new security issues: MAs are untrusted pieces of code that execute on possibly untrusted hosts over a possibly untrusted network used to migrate and communicate. Many flexible solutions to MA security problems have been recently investigated and deployed [6,7]. A still unresolved issue is to find, for any specific application, the most suitable trade-off between contrasting requirements, such as security levels and run-time performance.

We claim that both the necessary degree of interoperability and the proper level of security are key challenges

* Corresponding author. Tel.: +39-051-2093087; fax: +39-051-2093073.
E-mail address: pbellavista@deis.unibo.it (P. Bellavista).

0141-9331/01/$ - see front matter
PII: S0141-9331(01)00100-4

to widen the application opportunities of MA programming frameworks. These requirements should force designers of MA systems to provide a wide set of middleware services in order to simplify the deployment of secure and interoperable MA-based application components [8,9].

Another contribution in this special number discusses and proposes solutions for MA security issues [10]. This paper, instead, specifically focuses on middleware solutions for MA interoperability. Interoperability is greatly simplified by a large acceptance of standard interfaces and guidelines. In the area of distributed object-oriented systems, the Object Management Group (OMG) has proposed the common object request broker architecture (CORBA) [11]. CORBA permits to integrate distributed and heterogeneous application components, including legacy ones, independently of implementation languages and running operating systems.

The paper argues that MAs and CORBA are suitable and complementary technologies to provide services in an open and global scenario such as the Internet. We pursue the integration of the two technologies, and, in particular, the achievement of interoperability between heterogeneous MA platforms and between MAs and non-MA-based components via compliance with CORBA-based standards in the MA area.

Our ideas of interoperability and security have led to the implementation of an MA programming framework called SOMA[1] (secure and open mobile agents), implemented by using the Java 2 Platform [9,12]. From the interoperability point of view, the SOMA system provides application designers with facilities to simplify the implementation of agents acting as CORBA clients and servers. In addition, SOMA is compliant with the OMG mobile agent system interoperability facility (MASIF) that standardizes the basic functions of MA frameworks for agent management and transfer to external systems, whether MA-based or not [13]. MASIF is not the only CORBA-based standardization effort in the agent area: from a different perspective, the Foundation for Intelligent Physical Agents (FIPA) proposal mainly focuses on the definition of a standard architecture for agent platforms (APs) and of interoperable communication protocols/languages [14]. SOMA also provides a partial implementation of the FIPA specifications to answer the interoperability issues not covered by MASIF, such as message exchange between heterogeneous agents.

The SOMA interoperability service has a layered architecture consisting of three add-on modules (*CORBA C/S*, *MASIFBridge* and *FIPABridge*) that can be dynamically deployed only in the network nodes where (and when) their specific functionality is needed. The different interoperability modules are extensively exploited in a presented application scenario in the area of network, systems and service management, where they permit the interworking of SOMA agents, SNMP-compliant network elements, legacy resources/systems and non-SOMA MA platforms, such as Grasshopper [15].

[1] The SOMA platform is available at: http://lia.deis.unibo.it/Research/SOMA/.

2. CORBA-based solutions for interoperability in MA environments

MAs are computing entities that act on behalf of a principal (user, group, organization) and can autonomously migrate during the execution from one host to another one to continue their operations there. Agents can dynamically decide when and where to move, and can return results asynchronously, thus permitting to face critical situations where network connections are unreliable, bandwidth is scarce and even temporarily unavailable. In addition, MAs can work jointly and cooperate to provide distributed coordinated services.

These features suggest the adoption of the MA technology in several application areas, as more extensively described in Ref. [5]. For instance, systems management applications benefit from the fact that MAs can automatically perform management tasks on behalf of remote/disconnected administrators, possibly by migrating locally to managed resources. MA-based management components can report only significant events (e.g. the overcoming of even complex monitored thresholds) to the remote central authority without requiring a continuous exchange of monitoring information [12,16]. The management application domain particularly stresses the necessity of interoperation with heterogeneous components. The management of open systems/services imposes to monitor, control and possibly reconfigure distributed components independently of their implementation technology.

Several recent research activities recognize the importance of facing the complexity of service deployment in global distributed systems by providing layered architectures of standardized middleware components, called distributed processing environments (DPEs) [8,12,17]. These DPEs often use CORBA as the integration infrastructure.

CORBA provides a DPE where distributed objects can transparently interact according to the C/S model. CORBA simplifies the realization of C/S distributed applications, by hiding implementation and location of server objects from requesting clients. CORBA not only permits designers of distributed services to abstract from the details related to platform heterogeneity and object distribution, but also allows to integrate already implemented software components, independently of their possibly heterogeneous technologies. These legacy systems can be simply wrapped with an Interface Definition Language interface that describes their behavior. This possibility represents a non-negligible factor in the CORBA industrial success [18].

CORBA and MA technologies are different under several perspectives. The most notable one is *object mobility*. CORBA tends to assume objects allocated once and for

all at a fixed location before their registration at the object request broker (ORB), while MAs can dynamically and autonomously migrate during execution depending on run-time conditions.

Another difference is MA *location awareness*. MAs generally have visibility of their current locations and of the position of needed resources. This visibility is required to allow informed dynamic decisions about migration. At the opposite, CORBA tends to hide physical locations of server objects when answering client invocations. Obviously, the ORB should keep trace of the allocation of registered objects, but this information is typically transparent to client objects and application designers.

CORBA and MA technologies widely differ also in *diffusion*. CORBA has reached a wide acceptance and has a large base of compliant resources, systems and service components. This is demonstrated by the recent implementation of several bridging gateways to enable the interworking of Microsoft ActiveX/COM components with CORBA objects [19]. On the contrary, the novelty of the MA programming paradigm has led to a great variety of different and non-interoperable MA platforms.

The above considerations suggest that CORBA and MA can complement very well. In fact, a flexible framework for the provision of globally available Web services can benefit from both agent mobility and transparent remote agent interaction. It should provide different degrees of local/remote resource visibility. Finally, it needs to support the integration of heterogeneous and legacy service components via standard interfaces, in order to work in an open distributed environment.

The opportunity of integrating CORBA and MA is also demonstrated by the standardization efforts emerged to achieve interoperability between heterogeneous MAs. In fact, even if coming from different research communities and different scientific backgrounds, both the MASIF and FIPA proposals adopt CORBA as the standard bridge to overcome heterogeneity.

2.1. OMG MASIF

Interoperability among heterogeneous MA platforms requires identifying the aspects of the MA technology subject to standardization. The OMG has worked on the specification of MASIF, an agent interoperability standard, built within the CORBA framework, mainly to support agent mobility and management.

The idea behind the MASIF standard solution is to achieve interoperability between MA platforms of different manufacturers without enforcing radical platform modifications. In fact, MASIF does not require building new MA systems. Instead, its specifications should guide the design and implementation of an 'add-on' module to plug into already existing MA platforms. The standard includes CORBA IDL specifications that support agent transport, management and localization. It is worth stating that agent transport between different MA systems is not fully enabled through the given specifications. The transport capability requires additional mutual agreement between MA system implementers on the supported common format for agent exchange.

MASIF does not suggest standardization of local agent operations such as agent interpretation, serialization, execution and deserialization, because these actions are application-specific, and there is no reason to limit MA system implementations. Instead, MASIF only proposes standardization for agent and agent system names, for agent system types and for location syntax. It specifies two interfaces: the `MAFAgentSystem` interface provides operations for the management and transfer of agents, whereas the `MAFFinder` interface supports the localization of agents and MA systems in the scope of an administered locality.

As part of any MASIF-compliant agent system, the `MAFAgentSystem` object interacts internally with platform-specific services while it provides the associated CORBA interface to external users. In this way, it is possible to communicate with an agent system either in a MASIF-compliant way (using the `MAFAgentSystem` interface and the CORBA ORB) or in a platform-specific way (using platform-specific interfaces that may provide additional functionality not handled by MASIF).

2.2. FIPA

FIPA specifies the interfaces of the different components of APs to support the interaction with final users, other agents, non-agent software and the physical world. Being mainly proposed from the intelligent agent area, FIPA puts emphasis on the standardization of agent communication and proposes a dedicated Agent Communication Language (ACL) to support interoperable communication between heterogeneous FIPA-compliant agents.

The FIPA specification provides the normative framework for agents to exist and operate. It proposes the concept of an AP by offering three basic services. These services are namely the agent management system (AMS), the directory facilitator (DF) and the agent communication channel (ACC). Agents may offer their services to other agents and make their services searchable in a yellow pages manner by the DF. Registration on a DF is optional, while registering on the AMS is mandatory on any AP. Finally, the ACC enables communication between agents of possibly heterogeneous platforms with a message forwarding service. Reachability between platforms is gained by publishing the forward service over the CORBA ORB whose integration is considered mandatory for any FIPA-compliant MA platform. Agent messages are transferred via CORBA IIOP.

The AMS and DF services provide functionality similar to the MASIF `MAFAgentSystem` and `MAFFinder`. A peculiar characteristic of the FIPA standardization proposal

is the concept of agent communication by means of a special ACL. Agents have predictable behavior by common semantics defined in a common language interpretation, based on the concept of communication acts. For instance, the agent registration to an AMS is realized as a communication act of the action registration. This communication act clearly defines the roles of the agent and the AMS, and the reactions of each party are determined by the state of the AP. For instance, if the agent is already registered, the specification blocks any further registration, and a corresponding answer message must be yielded to the agent.

FIPA proposes the implementation of an ACC per agent system to forward ACL messages between agents. As platform local communication may find any MA implementation, the simplest solution for local communication between agents is obviously realized by the platform native communication protocol. FIPA-based inter-platform communication, that means communication between agents on different and possibly heterogeneous platforms, requires an implementation via a message forwarding service over CORBA.

3. The SOMA programming framework

We have designed and implemented the SOMA programming framework to offer a flexible MA environment with a rich infrastructure of middleware services. SOMA services include basic agent functions and more complex features suitable for the design and development of MA-based Web applications. The SOMA infrastructure is open also because it permits to extend dynamically the programming framework with new agent-based services, possibly built on the already provided functions.

Fig. 1 depicts the SOMA infrastructure consisting of two service layers. The lower layer, which provides the basic functionality for SOMA agents, includes:

- *The naming service*. It maintains and permits to access the information about the current state of any (possibly mobile) entity in the SOMA distributed middleware. A basic identification mechanism dynamically assigns tags to any entity in the system (agents, resources, service components and principals, i.e. users/organizations responsible for agent execution). These globally unique identifiers do not change even after entity migration. They are the basis for the realization of the SOMA naming service that puts together a set of different naming systems (e.g. DNS- and LDAP-compliant [20,21]).
- *The communication service*. It provides tools for communication and coordination between possibly mobile entities. When hosted in the same execution locality, agents interact by means of shared objects, such as blackboards and tuple spaces for tight cooperation. Otherwise, agents can perform coordinated tasks by exchanging asynchronous messages that are delivered also in case of migration of the target entity.
- *The migration service*. It supports the transport of one entity that requests to change its allocation. Entities capable of reallocation are represented by agents, which can move in the network either via MA native migration methods or via standard interfaces such as MASIF over CORBA IIOP. Reallocated entities can be traced also in their new locations by any entity in need of their services.

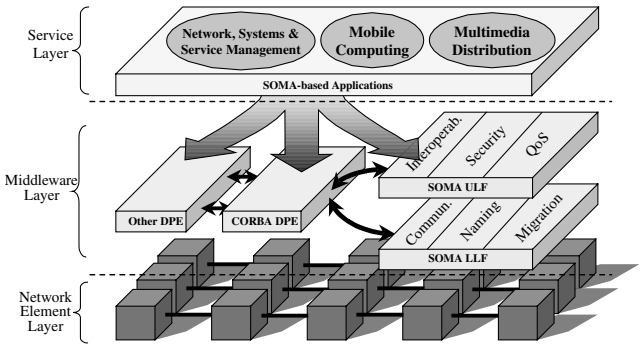

Fig. 1. The SOMA layered architecture of middleware services.

On the basis of this first level of middleware services, SOMA provides also an upper layer of MA-based general-purpose facilities:

- *The QoS monitoring and adaptation service*. The service is in charge of observing resource properties, from disk free space to effectively available network bandwidth, from CPU usage to heap memory allocated by any thread. This is achieved via the integration with the Java virtual machine profiler interface [22] and with platform-dependent monitoring modules via the Java native interface [23]. Any authorized mobile service can access the monitored properties, and, depending on this information, can decide a strategy suitable for adapting to the current environment conditions, without suspending service provision (e.g. for dynamically modifying the properties of distributed multimedia streams [12]).
- *The security service*. It aims to protect both MAs and hosting execution localities. Authentication is based on standard certificates and on a public key infrastructure. Authorization extends the Java standard mechanisms for access control. Secrecy is achieved by integrating the cryptographic libraries furnished by external providers. Integrity has required the development of MA-specific protocols for the protection of MAs from the execution environment.
- *The interoperability service*. It allows SOMA agents to interwork with existing software and hardware components via compliance with CORBA. SOMA implements the MASIF interface to enable the interaction of its agents with other MASIF-compliant MA platforms, and the FIPA ACC to support interoperable communication

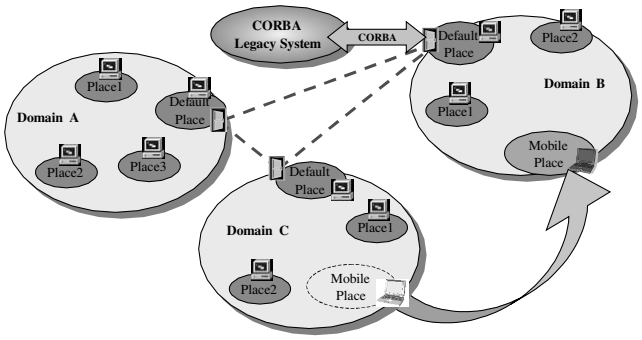

Fig. 2. SOMA locality abstractions.

of SOMA agents with agents of other platforms. A more detailed description of the SOMA interoperability service is presented in the following section.

In addition to providing this service infrastructure for MAs, SOMA offers locality abstractions to describe any kind of interconnected system, ranging from simple Intranet LANs to the Internet (see Fig. 2). Any node hosts at least one *place* for agent execution; several places are grouped into *domain* abstractions that correspond to network localities. In each domain, a *default place* is in charge of inter-domain routing functionality and integration with legacy components via CORBA. The *mobile place* is the locality abstraction used to support mobile devices: it enhances the normal place with specific features for automatic reconfiguration when changing domain of attachment [24].

Other details about the design and implementation of the SOMA programming framework are presented elsewhere [9,12] and are out of the scope of this paper that, in the following, specifically focuses on the SOMA interoperability service.

4. The SOMA interoperability service

Fig. 3 depicts the modular architecture of the CORBABridge component that implements the SOMA interoperability service. The different modules of the CORBABridge face four different interoperability challenges:

1. A SOMA agent may call external CORBA objects (SOMA agents as CORBA clients).
2. A SOMA agent may publish its interface to the CORBA ORB (SOMA applications as CORBA servers).
3. SOMA can manage and accept MASIF-compliant heterogeneous agents, and SOMA agents can be managed and transferred towards MASIF-compliant heterogeneous MA systems (interoperability between SOMA, other MASIF-compliant MA platforms, and CORBA management components).
4. A SOMA agent may send/receive messages to/from any FIPA-compliant AP via the ACC forward service (interoperable communication between heterogeneous FIPA agents).

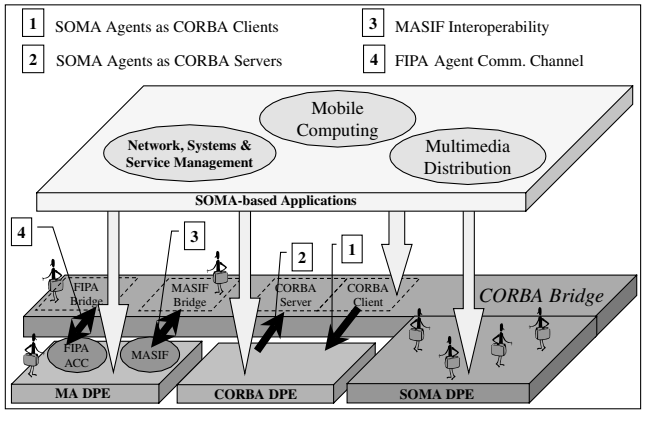

Fig. 3. The modules of the SOMA interoperability service.

The CORBA C/S module provides the first two interoperability features: agents can play the role of CORBA clients, and can register as CORBA servers to offer access points to applications outside the SOMA system. There is no conceptual problem in an MA that registers as a CORBA server. However, we currently grant this possibility only to SOMA agents that do not migrate during their lifetime (*stationary agents*) to avoid the overhead of registering/unregistering to the CORBA ORB (and possibly to the CORBA naming and trading services) at any migration.

The third feature is a more complex issue and SOMA addresses it via compliance with the MASIF standard (MASIFBridge module). Any external system can control remote agents of a MASIF-compliant MA platform via the `MAFAgentSystem` interface: MASIF defines methods for suspending/resuming/terminating agents and for migrating agents from one MA platform to another one. The interoperation is significant only when the two interworking systems present a compatibility base, that is the same implementation language, or compatible externalization mechanisms. Agent tracking functions permit the tracing of agents registered with `MAFFinder`, introduced to provide an MA name service, because the CORBA naming service is not suitable for entities that are intrinsically and frequently mobile.

At the moment SOMA agents can communicate via proprietary mechanisms and protocols, but can also decide to exploit the CORBA middleware to coordinate via shared CORBA objects. In addition, because agent communication is outside the MASIF scope, we have decided to integrate an additional interoperability module (FIPABridge) that mainly implements the FIPA ACC to support interoperable communication between heterogeneous agents. The FIPABridge ACC is available as a place facility that SOMA agents exploit to convert messages into the corresponding ACL format and vice versa, with an approach similar to the one followed in the Jade project [25]. The FIPA AMS and

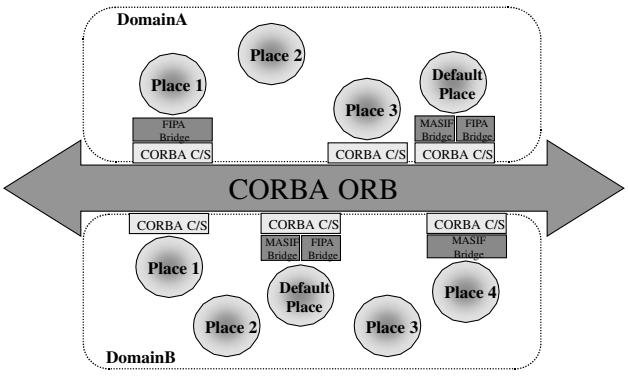

Fig. 4. The SOMA interoperability service as the composition of plug-in modules.

DF facilities are implemented by mapping them into the analogous functionality for agent management and registration already available, respectively, in the SOMA `MAFAgentSystem` and `MAFFinder` modules.

The SOMA programming framework provides a modular implementation of the CORBABridge, with the different specific interoperability modules that can be even installed at run-time. At their creation, SOMA places are configured with the minimal SOMA distribution, e.g. without any support for interoperability. System administrators may decide dynamically to upgrade a subset of SOMA places with the CORBA C/S module (see Fig. 4). For instance, an administrator can specify a management policy to command the automatic installation of the CORBA C/S in all the SOMA places that include legacy systems with CORBA interfaces (registered either in the global directory or in the local discovery services [24]). Specialized installation agents are able to extend SOMA places with the CORBA C/S component, without imposing any service suspension.

In addition, other SOMA agents are able to mount dynamically also the MASIFBridge and the FIPABridge modules on any SOMA place that already hosts the CORBA C/S component. On the one hand, the MASIFBridge extends the SOMA default places that need to interact with other non-SOMA MA platforms compliant with MASIF. Only default places can host the MASIFBridge, since MASIF implementation increases significantly the code size and it is natural to provide it only in the place responsible for inter-domain routing functionality. On the other hand, the FIPABridge is added only where administrators decide to enable interoperable communication between heterogeneous agents (sending/receiving FIPA-compliant messages).

5. Interoperability solutions at work: SOMA for heterogeneous network/systems management

Since the first MA research activities, the domain of network, systems and service management has attracted much interest because of the potential advantages of the MA technology application, e.g. in terms of agent autonomy, asynchronicity and capacity to migrate locally to the managed resources. In addition, recent management applications tend to provide solutions suitable for geographically distributed systems, which are open and intrinsically heterogeneous. Therefore, the management domain has represented an ideal workbench to put our MA interoperability solutions at work.

The simple network management protocol (SNMP) is certainly the most diffused and simple solution for the monitoring and control of network elements. MAs can significantly improve the performance of SNMP management, in terms of both reducing network traffic and of minimizing the time to trigger control actions [26]. However, management applications often need to control also heterogeneous legacy systems that do not host the execution of SNMP agents. In this case, a viable and pragmatic solution is to encapsulate legacy components into CORBA wrappers, thus providing a CORBA-based point of access for their control. In addition, the CORBA integration is often the approach followed to manage network elements that adopt management protocols and models different from SNMP. A notable example is the case of components supporting the OSI common management information protocol (CMIP); several vendors have implemented gateways to integrate CMIP-compliant resources via CORBA interfaces [27].

To address these issues, we have designed and implemented a SOMA-based management tool that is able to monitor and control several types of networked systems and service components. The main idea is to exploit agent mobility to migrate intelligent monitoring entities locally to the resources to be managed. Fig. 5 depicts a usage scenario where our management tool exploits a hierarchical organization of SOMA agents to aggregate monitoring information about SNMP elements, CMIP components, CORBA legacy systems and other resources belonging to non-SOMA locality domains (in particular, Grasshopper regions). The distribution and organization of manager/slave agents is similar to the one presented in Ref. [26].

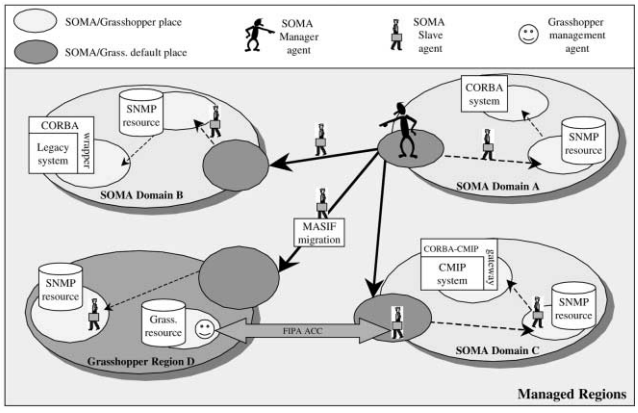

Fig. 5. The SOMA-based management application at work on heterogeneous distributed resources.

We have started our work with the implementation of SOMA agents that use the SNMPv3 protocol for monitoring SNMP-compliant components located in SOMA domains. These components are visible because of their registration to the SOMA naming service; management agents can search the naming service and, on the basis of the query results, can migrate either locally or in proximity to the SNMP-compliant devices to control.

Our goal is to provide the widest possible coverage of heterogeneous resources to manage, without addressing only SNMP-compliant systems. Thus, our tool exploits SOMA components to manage:

1. networked resources that publish a CORBA interface;
2. networked resources located in regions under the control of non-SOMA MA platforms.

The first type of resources can include legacy systems encapsulated into CORBA wrappers, CORBA native components and CMIP-compliant systems accessed via the CORBA/CMIP gateway integrated in the SOMA-based management tool [12]. Fig. 5 shows SOMA agents that move to the CORBA resources and monitor/control them by exploiting the CORBA C/S interoperability module.

The second case addresses the management of components belonging to any MA platform, provided that the platform is conforming to either the MASIF or the FIPA specifications. Fig. 5 depicts a scenario that includes one region hosting the Grasshopper MA system, which is compliant to both MASIF and FIPA. The Grasshopper resources to be managed are registered to the `MAFFinder` object published by the Grasshopper domain. On the one hand, Grasshopper components can be managed by migrating SOMA agents to their locality. This is possible via the receive_agent( ) method of the `MAFAgentSystem` object of the target Grasshopper domain. On the other hand, our management tool permits also to delegate monitoring and control operations to known Grasshopper management agents running in that locality. SOMA agents can remotely exchange monitoring data and control commands by exploiting the interoperable communication mechanisms provided by the FIPA ACC and implemented in SOMA by the FIPABridge module, as illustrated in Fig. 5.

6. Emerging trends in MA interoperability

Despite the number of approaches in the design and implementation of MA systems, some trends are clearly emerging, as presented in Ref. [5]. Interpreter-based programming environments represent the common basis for most MA frameworks. In particular, Java is frequently chosen not only for portability, fast prototyping and easy integration with the Web, but also for its security features, such as type checking and fine-grained flexible control of resource access. In addition, security has attracted deep interests in the MA community: several MA platforms provide at least a subset of the security services required to implement MA-based applications in open untrusted environments.

Interoperability, instead, has not received yet the due attention, and a definite common trend on how to achieve it is still unclear in the MA research area. Some MA platforms provide wide accessibility to their MA-based applications and to their development/management tools. For instance, *Voyager* [29] permits agents to be activated via an applet interface; it additionally allows access from and to external objects via CORBA. Similarly, SOMA provides a Web interface to securely launch, monitor and control agent state/position in any SOMA place that activates applet-control over SSL communication. *IBM Aglets* [30] provide an additional package for running an Aglet context (the analogous of a SOMA place) on an HTTP-browser.

Accessibility, however, is only the first basic step towards complete interoperability, to enable full interaction between heterogeneous systems, whether MA-based or not, and to support cooperation between all existing components in the provision of flexible network-centric services.

A few MA frameworks have already approached the issue of interoperability with other systems and applications. Those proposals promote CORBA as the standard integration technology. For instance, *Jumping Beans* [28] implement their agents by extending standard CORBA objects with proprietary mobility mechanisms. Whenever a Jumping Beans agent decides to migrate, however, it is forced to move to a single server that becomes a centralization point, thus severely limiting the scalability of the system. Finally, *Grasshopper* [15] is today the commercial MA platform that most completely addresses interoperability: in addition to Web connectivity, it allows interaction with non-MA-based objects via CORBA, and is currently the only MA programming framework fully compliant to both MASIF and FIPA.

7. Conclusions and current work

The design, implementation and deployment of Internet-centric services motivate the adoption of new flexible programming paradigms, capable of moving data and behavior dynamically. The paper integrates two solution directions, MAs and CORBA, which are sometimes considered diverging and in contrast.

On the one hand, global network systems force to distribute responsibilities to a plurality of components, able to operate autonomously and in need of coordinating flexibly with each other. In this context, the MA technology represents a clean and uniform approach to a wide spectrum of application areas, from network and systems management to mobile computing, from distributed information retrieval to electronic commerce. On the other hand, in large and open intra-/inter-corporate networks, which are intrinsically heterogeneous, CORBA is already the accepted integration

technology to deal with network, platform and programming language diversity.

The integration of MAs with CORBA and CORBA-based standards plays a central role to achieve interoperability by providing standard bridges among proprietary MA implementations. Along this guideline, the SOMA interoperability service offers a wide spectrum of modular solutions to overcome the different barriers that limit the current applicability of the MA technology. The CORBABridge modular implementation demonstrates not only the feasibility of the MA–CORBA integration, even in terms of introduced overhead [9], but also the possibility to provide extremely flexible and configurable interoperability supports, composed by layers of dynamically installable modules.

In addition to the presented utilization of SOMA agents in network, systems and service management, we are working on other application domains that particularly stress the MA interoperability issues. We have implemented a SOMA-based middleware prototype to support user, terminal and resource mobility, by reconnecting transparently mobile entities to their needed resources and services, independently of current location and implementation heterogeneity [24]. Finally, we are developing an MA-based information retrieval service where agents can autonomously and locally filter distributed heterogeneous museum data, and can present query results in an XML-based interoperable format.

Acknowledgements

Investigation supported by the Italian Ministero dell'Università e della Ricerca Scientifica e Tecnologica (MURST) in the framework of the Project 'QoS Infrastructure for Web-based Multimedia Services with Heterogeneous Accessibility', the Italian 'Consiglio Nazionale delle Ricerche' in the framework of the Project 'Global Applications in the Internet Area: Models and Programming Environments' and by the University of Bologna (Funds for Selected Research Topics: 'An Integrated Infrastructure to Support Secure Services').

References

[1] A. Fuggetta, G.P. Picco, G. Vigna, Understanding code mobility, IEEE Transactions on Software Engineering 24 (5) (1998).
[2] J.W. Stamos, D.K. Gifford, Remote evaluation, ACM Transaction on Programming Languages and Systems 12 (4) (1990).
[3] D.B. Lange, D. Milojicic (Eds.), First International Symposium on Agent Systems and Applications and Third International Symposium on Mobile Agents (ASAMA'99) IEEE Computer Society Press, Palm Springs, CA, 1999.
[4] M. Wooldridge, K. Decker, Special section on agents on the Net, IEEE Internet Computing 4 (2) (2000).
[5] A.R. Tripathi, T. Ahmed, N.M. Karnik, Experiences and future challenges in mobile agent programming, Microprocessors and Microsystems 25 (2) (2001).
[6] G. Vigna (Ed.), Mobile Agents and Security Lecture Notes in Computer Science, vol. 1419, Springer, Berlin, 1998.
[7] U.G. Wilhelm, S.M. Staamann, L. Buttyan, A pessimistic approach to trust in mobile agent platforms, IEEE Internet Computing 4 (5) (2000).
[8] J. Bolliger, T. Gross, A framework-based approach to the development of network-aware applications, IEEE Transactions on Software Engineering 24 (5) (1998).
[9] P. Bellavista, A. Corradi, C. Stefanelli, Protection and interoperability for mobile agents: a secure and open programming environment, IEICE Transactions on Communications E83-B (5) (2000).
[10] N. Dulay, R. Montanari, C. Stefanelli, Microprocessors and Microsystems 25 (2) (2001).
[11] CORBA/IIOP Rev 2.2, OMG Document formal/98-07-01, Object Management Group, February 1998. http://www.omg.org/library/.
[12] P. Bellavista, A. Corradi, C. Stefanelli, An integrated management environment for network resources and services, IEEE Journal on Selected Areas in Communication 18 (5) (2000).
[13] Mobile Agent Facility Specification, Joint Submission supported by Crystaliz Inc., General Magic Inc., the Open Group, OMG TC Document orbos/98-03-09, GMD FOKUS, IBM Corp., September 1998. ftp://ftp.omg.org/pub/docs/orbos/98-03-09.pdf.
[14] FIPA 2000 Specifications, Foundation for Intelligent Physical Agents. http://www.fipa.org/.
[15] IKV++, Grasshopper: the Agent Platform. http://www.grasshopper.de/index.html.
[16] A. Karmouch (Ed.), Special Section on Mobile Agents, IEEE Communications 36 (7) (1998).
[17] TINA DPE Architecture, Telecommunications Information Networking Architecture Consortium, March 1998. http://www.tinac. com/.
[18] K. Seetharaman (Ed.), Special Section on CORBA, ACM Communications 41 (10) (1998).
[19] Web + Object Integration, Object Services and Consulting Inc., August 2000. http://www.objs.com/survey/web-object-integration.htm.
[20] P. Albitz, C. Liu, DNS and BIND, 3rd ed., O'Reilly & Associates, 1998.
[21] T. Howes, M. Smith, LDAP: Programming Directory — Enabled Applications with Lightweight Directory Access Protocol, Macmillan, New York, 1997.
[22] Java Virtual Machine Profiler Interface (JVMPI), Sun Microsystems. http://java.sun.com/products/jdk/1.3/docs/guide/jvmpi/jvmpi.html.
[23] R. Gordon, Essential Java Native Interface, Prentice-Hall, Englewood Cliffs, NJ, 1998.
[24] P. Bellavista, A. Corradi, C. Stefanelli, A mobile agent infrastructure for terminal, user and resource mobility, in: Proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS 2000), Honolulu, HI, April 2000.
[25] Jade, CSELT. http://sharon.cselt.it/projects/jade/.
[26] D. Gavalas, D. Greenwood, M. Ghanbari, M. O'Mahony, Mobile software agents for decentralised network and systems management, Microprocessors and Microsystems 25 (2) (2001).
[27] The UHC CORBA/CMIP Gateway, UH Communications ApS. http://www.uhc.dk/.
[28] Jumping Beans, Ad Astra Engineering Inc. http://www.Jumping Beans.com/.
[29] Voyager, ObjectSpace. http://www.objectspace.com/.
[30] Aglets SDK1.1, IBM Japan. http://www.trl.ibm.co.jp/ aglets/.

Paolo Bellavista received his Laurea in electronic engineering from University of Bologna, Italy, in 1997. He is currently pursuing a PhD in computer science engineering at the Department of Electronics, Computer Science and Systems of the same university. His research interests include distributed computing, distributed objects, mobile agents, network and systems management, adaptive multimedia systems and distance learning. He is a student member of the IEEE, ACM and AICA (Italian Association for Computing).

Antonio Corradi received his Laurea in electronic engineering from the University of Bologna, Italy, in 1979, and his MS in electrical engineering from Cornell University in 1981. He is currently 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 is a member of the IEEE, ACM and AICA.

Cesare Stefanelli received his Laurea in electronic engineering from the University of Bologna, Italy, in 1992 and the PhD degree in computer science in 1996. He is currently an Associate Professor of operating systems at the University of Ferrara. His research interests include distributed and mobile computing, mobile code, network and systems management, network security. He is a member of the IEEE and AICA.