From Intelligent Agents towards Management by Request

Raul Oliveira, Jacques Labetoulle

Corporate Communications Department
Eurecom Institute
06904 SOPHIA ANTIPOLIS CEDEX
France

Telephone: 33 4 93002653
FAX: 33 4 93002627

Abstract

In this article we introduce a management framework that allows the participation of major actors in the distributed computing environment(user applications and service providers) into network and systems management. This participation is realized by creating mechanisms that allow actors to express their requirements and capabilities to the management infrastructure. To make possible this interactive behavior between these actors and the management infrastructure, Intelligent Agents are proposed as management front-ends in charge to accept what we call management requests. This article also intends to present both the information model and the execution environment of this framework.

1 Introduction

Improving the network intelligence has been the aim of several research projects. Different approaches have been taken ranging from: experts systems, case based reasoning (CBR) or even model based reasoning (MBR) [3]. In our opinion they have not been completely successful, mostly because these knowledge based techniques were often used to capture environment dynamics, while being completely unaware of user behaviors or user needs. Nevertheless, rewriting these approaches in a distributed fashion does not seem to be the solution. For us the right way to improve network intelligence is to find mechanisms to increase the awareness of: user requirements, service provider capabilities, current service configuration and dependencies. These information, rather than being centralized, must be distributed by the entities directly responsible for them, and provided to the network and systems management infrastructure as they become needed.

Most of the knowledge expressed through rules in expert systems, cases in CBR systems or models for MBR systems, could be explicitly provided by network actors through well designed information structures. Of course, such an information model must be designed in a way that could be easily enhanced during normal operation, without major changes on current management system structure.

Another form of intelligence in networks is expressed by management applications ability to share knowledge, goals and tasks. This is of special importance when cooperation takes place across domain boundaries. At present, when two domains need to exchange management information, it is necessary to impose distinct management roles to the management applications in charge on both domains. In fact, these applications need a management role to cooperate, distinct from the current available roles (manager and agent), that are more appropriated to client server relationships.

In this way, a management application in a domain should be able to demand management operations, define management goals or ask for cooperation, in a certain task, to other management applications in other domains. Today, such as specified in TMN[4], each domain should offer an interface [1], where foreign applications can invoke the desired management operation. While invoking these operations one of the applications is in a manager role while the other is in an agent role. In our opinion this constitutes a very primitive communication paradigm, inherited from the Element and Network TMN layers, but clearly not suited to other layers of the TMN model.

[1] Interface X preview the use of CMIP/CMISE

2 Aware Management

Awareness is probably the most important evidence of intelligence in a networked environment. For us aware management means that the management infrastructure should be aware of users requirements or behavior profiles. In our opinion, awareness have two major benefits:

- accepting user requirements allows to selectively monitor services and resources in order to maintain the user QoS the closest as possible to user aims.
- having the "user" behavior profiles allows that any management operation could be adapted to networked environment dynamics.

Management by Request

Usually management operations are determined offline by network or system operators who know the environment they have to manage. They choose among what seems more critical to monitor in order to maintain a good overview of the global distributed system. But, most of the time they end up proceeding reactively with the alarms being received by the management applications they control. It is evident that they cannot handle users individually and so they are not able to follow environment dynamics.

In fact, in our management framework we propose another management paradigm: The Management by Request.

- "Each application or user entering the management environment will publish its requirements in a service oriented form, specifying the QoS required for each of those services. In fact these users or applications will be implicitly requesting the management of these services with the appropriate granularity for the tasks they are performing".

Management by Request is a management paradigm that requires autonomous applications to handle the management requests and react appropriately to them, by creating on the fly the activities satisfying those requests. Intelligent Agents following architectures such as "Motivated Agency" [5] or BDI [10] (Belief, Desire and Intentions) are the kind of solution we adopted. IAs essentially create goals in reaction to certain environment conditions (user requirements). Because there will always be situations in any real world domain that can never be predicted – can only be reacted to when they occur [6](when the user states that he is about to use some services and resources).

3 Management Information Model

To define our management information model we adopted some of the architectural concepts proposed for TMN and defined in the M.3000 recommendations. TMN calls for a "logical layered architecture" in which management tasks are broken in subsets and then partitioned into layers. The layers are logical because they may not correlate exactly with the physical implementation [4].

Following the TMN architecture our management framework is clearly identified with the service layer. This is the layer where all network actors can be grouped around a common interest: the **service**.

This management layer is specially suited to rise the abstraction level, and therefore allowing all actors, directly or indirectly concerned with management, to easily share information about service utilization. Because each actor has a different view of the management environment and namely the services, it is interesting to consider an information model organized in four different planes.

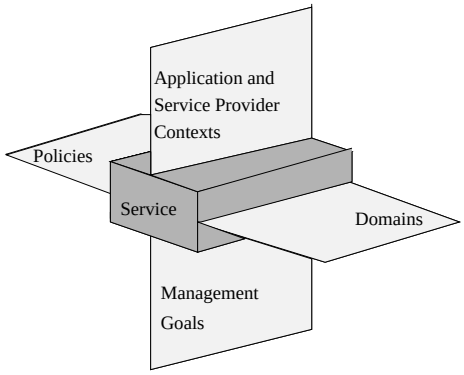

Figure 1:Service oriented four plane information model

The smallest information unit, understandable by all the actors, on the management environment, is the service [2], regardless of the technology used to build the service or to perform the service management.

The management information model is hence designed in a service oriented way, and structured in four distinct planes: requirements/capabilities, policies, domains, and goals. Hence, while participating in the management, all actors have the service as a reference unit of information. The service constitutes therefore the commonality between all the actors.

[2]Hereafter when we say service we will be referring to the user oriented service. Which must not be confused with service primitives of base protocols

This approach lends to a framework where it is possible to separate management targets from service implementation details. Such as management protocols, structures of management information and the definition of managed objects.

For applications and service providers the requirements and capabilities are expressed through contexts with service granularity. Intelligent Agents in the same way, and besides their ability to process contexts from application and service providers, will create their own management goals in terms of services. In what concerns managers, they will also inform Intelligent Agents about applicable policies in service oriented way.

Applications and Service Providers Contexts

In [8] we chose contexts to let applications inform Intelligent Agents of their requirements, while organizing these contexts in a service oriented way. For each service the application add the QoS dimensions [1] that match the applications requirements. Each of these dimensions could be made up of several QoS domains which finally are composed of sets of attributes, that characterize the QoS required for the service. The number of flavors for QoS dimensions can be high, as well as the number of QoS domains of each dimension. So applications must publish their service requirements as a sequence of needed QoS dimensions and for each QoS dimension a sequence of QoS domains.

In most network environments the application will also need to specify the resources associated with the service and as well as the identification of the service provider.

For service providers, we also preview the use of contexts, in order that the management infrastructure could be informed about who is offering the services, and what is the offered QoS capability. These contexts are structured in the same form as they were for applications. For instance, a service provider declare the QoS dimensions that are useful to know the capabilities of the service provider and help in monitoring its behavior and health.

- *Application Context{*
  - *Application Context Header{*
    - *Application name*
    - *Application hostname*
    - *Application priority*

    }
  - *Global Application QoS{*
    - *Sequence of QoS Dimensions*
    - *Sequence of active QoS Dimensions*

    }
  - *Service QoS Specification {*
    - *Sequence of Service sub-contexts*
    - *Sequence of active sub-contexts*

    }

  }
- *Service Sub-context{*
  - *Service Context Header{*
    - *Service name*
    - *Service impact on application behavior}*
  - *Sequence of QoS Dimensions*
  - *Sequence of active QoS Dimensions*
  - *Sequence of service Resources}*
- *QoS Dimension{*
  - *QoS Dimension name*
  - *Sequence of QoS Domains*
  - *Sequence of active QoS Domains*
  - *Sequence of dimension specific Resources }*
- *QoS Domain{*
  - *QoS Domain name*
  - *Sequence of Domain Attributes }*

Figure 2: Context organization and service oriented QoS specification .

Management policies

Obligation and authorization are the two most important policy types in reference frameworks such as IDSM and SysMan [2]. These policies can be used in a negative or positive sense, for obliging or deterring, and authorizing or prohibiting respectively. We think that these two types of policies are not enough when dealing with Intelligent Agents. Intelligent Agents rather than being obliged or authorized should preferably be motivated, which is more appropriate [7].

Since we have accepted these three policies flavors as sufficient, we explain why we need these policies to deal with intelligent Agents.

Obligation policies are useful when we need to request an IA to perform one or more management actions over determined targets. So they are management orders that a manager send to an IA, and which are accomplished without any other interpretation.

Motivation policies will be used to create propensity to execute some type of management tasks, as soon as they are required. Execution tasks decision

is up to the Intelligent Agent based on the perception it has from the networked environment.

Unless any reason we do not foresee now, we will not use authorization policies to directly influence Intelligent Agents behavior. In our opinion, as soon as an IA receives obligation or motivation policies it is already and implicitly authorized to perform the associated management operations. However, authorization policies are still of paramount importance to specify which network actors, whether in a management or user role, belonging to local or foreign domains, can request management operations from an Intelligent Agent.

As we made for application or service providers contexts in our framework we try to avoid considering particular features of the different management environments. So again we will specify management policies oriented to service abstraction instead of defining policies applicable directly over real managed objects.

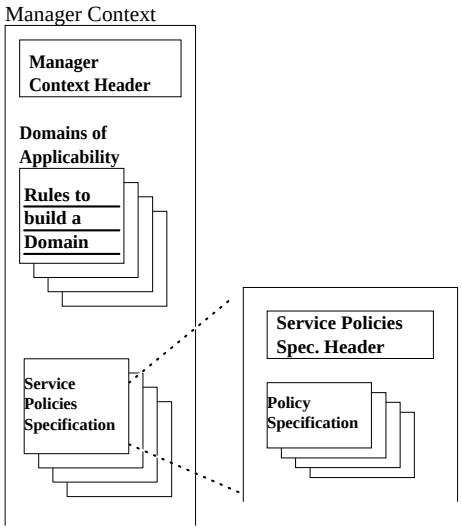

Figure 3:Organization and structure of manager contexts

As we stated in [8], management policies and domain specifications will be carried from managers to Intelligent Agents in information structures called Manager Contexts. Figure 3 shows how these manager contexts are organized and structured.

Intelligent Agents goals

Intelligent Agents goals represent the current Intelligent Agent management intentions. Management Goals are, therefore, a possibility that managers or network operators dispose to monitor IAs current management activities. So they should be available to managers or others IAs on demand.

Intelligent Agents Goals are a high level semantics to describe management operations, independent of service implementation technology and service management paradigms.

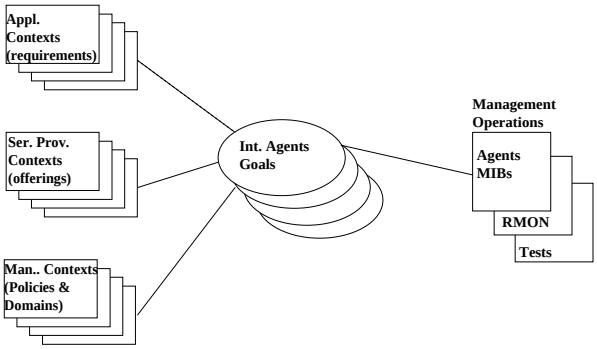

Figure 4:Intelligent Agent goals as relationships between contexts and management operations

A management goal is created to verify, guarantee or observe if user requirements are being satisfied. Management goals are created expressing dependencies on requirement and offering contexts, and it is this aggregation that allows to create management goals that take in account networked environment dynamics. Intelligent Agent management goals are in fact relationships between contexts and management operations.

IA goals are characterized according to its lifetime (ephemeral, short-term, medium-term, long-term), or according to the type of activity (calculating metrics, checking, monitoring, obtaining baselines, health analyzing and testing). These activities should be built up of standard management operations: read/write MIBs variables, programming RMONs and service testing (by behaving as a common service client).

4 Management execution environment

The execution environment that we propose in our framework creates some communication axes that did not exist before. Such are the communication between applications or service providers and the management infrastructure, and between management applications in an equivalent role (peer-to-peer communication).

Contexts publication

Publishing is the way we choose to make applications or service providers inform the management

front-ends [3] of their requirements or offer capability. This type of interaction was preferred to the traditional discovering process, since discovering entails a waste of bandwidth and processing resources, to obtain the equivalent information.

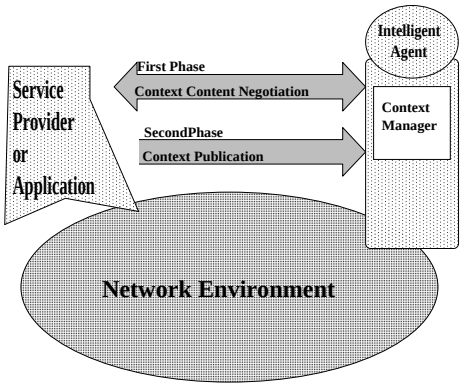

Figure 5:Context content negotiation and publication.

Applications or service providers can specify in their contexts a variable number of services, QoS dimensions and QoS domains.

It is possible that the Intelligent Agent facing an application does not have the knowledge to handle all the application context content. To avoid situations where an IA is not able to understand application contexts, the first step is a negotiation act between the application and the Intelligent Agent. The knowledge of which information can be handled by the Intelligent Agent, is then used by the application to define and publish its context. From the Intelligent Agent side the negotiation process is an opportunity to obtain the service knowledge before the application publishes its context.

The publication mechanism is used for all forms of contexts that an IA is able to process: application requirements contexts, service providers capabilities contexts and manager contexts.

Dynamically downloaded code and remote execution

The flexibility introduced for the publication of service requirements in contexts, may require a periodic update of IA software. For example when an application requires a new service to be managed, that was unknown to the IA until now. The same when the basic service management knowledge is already available on the agent, but it is necessary to handle a new service QoS dimension or domain. In this case the IA need to load to itself the required code.

[3] in [8] this was how the Intelligent Agents were named in the proposed management infrastructure

For remote execution mobile code can be used, for instance, when the Intelligent Agent knows that there is a problem on a service provider, but it does not have any means to solve the problem remotely through traditionally management interfaces. In this case the IA sends a mobile Agent to the service provider host to solve the problem locally [4]

Agents Cooperative Work

In this framework the IAs are the center of our execution environment. They receive requirements and capabilities, and according to domains of action and policies specified by managers, they create their own management goals. These goals are whether to achieve locally in the domain, or can be delegated to other IAs in other domains. Depending if requirements that implied those goals, specify services provided on those domains. It is up to each IA to discover other IAs able to monitor and guarantee the quality of service the users are requesting.

To allow such cooperation between IAs it is necessary to find a language and associated protocol support. As we stated before, the manager/agent paradigm is not suited to this type of cooperation. Also notice that CMIP protocol service primitives such as M-GET, M-SET and M-CREATE are not the most expressive to intelligent cooperation. By adopting KQML [9] Agent Communication Language (ACL) we expect to benefit from performatives carrying by themselves high semantic content such as: ask, tell, deny, subscribe, advertise, recruit, achieve, etc. These performative types allow an IA to precisely demonstrate its intentions to other IAs. Besides the powerful semantic level expressed by KQML performatives, there are other facilities, on this ACL. Such as the possibility to chose among various content languages, organize communication acts around several ontologies (allows the organization around services, management functionalities, and management areas). Finally this ACL does not use the one-to-one relationship between management applications. It enables one Agent to "broadcast" or multicast messages to various Intelligent Agents.

5 Conclusion

Management by Request can be considered a well-suited paradigm for management in a future, where

[4] This execution paradigm can be widely used, independently of operating system, being an advantage in when compared to Remote Shell UNIX paradigm.

a close collaboration between users and the network and distributed systems infrastructure is expected. We presented what we understood by such an approach, and we designed a management information model for the management layer where we think this framework is suited for. In the same way, because this approach would entail an intelligent management infrastructure, we proposed the adoption of intelligent communication paradigms, such as Agent Communication Languages, between management applications, and in this case Intelligent Agents. Finally, to cope with the fast evolution on management infrastructures and respective applications, we propose two forms of mobile code for such an execution environment.

References

[1] TINA Consortium. Quality of Service Framework, Draft TINA Report, November 1994.

[2] Domain and policy service specification. In K. Becker, U. Raabe, M. Sloman, and K. Twidle, editors, *IDSM Deliverable D6, SysMan Deliverable MA2V2*, October 19 1993.

[3] Shri K. Goyal. *Network and Distributed Systems Management*, chapter 21, Artificial Intelligence in Support of Distributed Network Management, pages 539–577. Addison-Wesley Publishing Compagny, 1994.

[4] Kirk Shrewsbury J. An introduction to TMN. *Journal of Network and Systems Management*, 3(1), 1995.

[5] Timothy J., Norman, and Derek Long. Alarms: An implementation of motivated agency. In Michael Wooldridge, Jorg P. Muller, , and Milind Tambe, editors, *Intelligent Agents II - Agent Theories, Architectures, and Languages*, pages 219–234, Montreal, August 19-20 1995. Springer.

[6] Timothy J., Norman, and Derek Long. Goal creation in motivated agents. In Michael Wooldridge and N. R. Jennings, editors, *Proceedings of the 1994 Workshop on Agent Theories, Architectures, and Languages*. Springer, 1995.

[7] Raul Oliveira and Jacques Labetoulle. Intelligent agents a new management style. In *Accepted to Distributed Systems and Operations Management Workshop - DSOM'96*, L'Aquila, Italy, October 1996.

[8] Raul Oliveira, Dominique Sidou, and Jacques Labetoulle. Customized network management based on applications requirement. In *Proceedings of the First IEEE International Workshop on Enterprise Networking - ENW '96*, Dallas, Texas, USA, June 27 1996.

[9] G. Wiederhold T. Finin. An overview of KQML: A Knowledge query and manipulation language, 1991. Available through the Standford University Computer Science Dept.

[10] M. Wooldridge. A logic of bdi agents with procedural knowledge. In J. L. Fiadeiro and P.-Y. Schobbens, editors, *Proceedings of the Second Workshop of the MODELAGE Project*, Sesimbra, Portugal, January 15-17 1996.