Intelligent Agents In Network Management A State-of-the-art*

Morsy M. Cheikhrouhou, Pierre Conti and Jacques Labetoulle
Corporate Communications Department
Eurecom Institute
BP 193, 06904 Sophia Antipolis Cedex
Tel: + 33 4 93 00 26 94, Fax: +33 4 93 00 26 27
email: cheikhro, conti, labetoul@eurecom.fr

Abstract

Networks will soon be the key stone of all industries. However, the available NMSs are not adapted to the wide spectrum of network installations and configurations. The introduced new technologies, e.g. CORBA, seem to be not sufficient to solve the problems of complexity, cost and scalability. Different studies are on the way to distribute the intelligence to the different network components as a logical answer to these issues. Among them, the Intelligent Agent paradigm seems to be the most promising solution. This paper -after a short reminder on the NMS issues and an introduction to the IA concepts- presents a synthesis on the current research on the IA for NM. First, it discusses the management by delegation paradigm. Afterwards, it handles the different ways mobile agents are suggested for NM frameworks. The remainder of the state of the art addresses Intelligent Based NM approaches according to the agent architecture and to cooperation aspects. Finally, the paper presents the perspectives and the potential benefits of IAs in NM.

Keywords : Network Management, Intelligent Agents, Mobile Agents, Management by Delegation

1 Introduction

In the near future, all the computers will be inter-connected. We will find networks not only in world wide company as it is the case now, but also inside intelligent buildings and even inside houses to drive toasters or washing machines [1]. Like other systems, these networks need to be controlled and managed. Furthermore, it is obvious that personal networks (e.g. house networks) or small enterprise networks, would not require the same Network Management Systems in terms of complexity, cost and necessary resources. Actually, if almost all networkable devices are manageable (mostly via SNMP), only few Network Management plateforms exist, all of them being designed for huge networks. Looking for scalable, flexible and economic solutions, the emerging Intelligent Agent paradigm seems to be a solution. But if hundreds of papers on Intelligent Agents are available through proceedings or on the web, showing the incredible mess of the Agent World, only a few of them have dealt with the use of Intelligent Agent for Network Management. This situation is now changing and the network management community is realizing that "As we move further and further into the information age, any information-based organization which does not invest in agent technology may be committing commercial hara-kiri" [2].

In this paper, we present a State of the Art on the use of Intelligent Agents in Network Management. The report can be viewed as comprising three parts. The first part consists of two

*This paper has been written within the framework of the project F&E 391. This project is financially supported by SwissCom

sections where the first section (section 2) gives a review of the network management principles and issues. The second section (section 3) gives an introduction on the Intelligent Agent principles. It states the properties that IAs might have as a way to percieve what the IAs are.

The second part presents the researches on IAs in the Network Management domain. The first section of this part (section 4) is dedicated to present an important trend in NM called the Management by Delegation. It focuses on the works that have used the concepts of MbD to build Intelligent Agents, and discusses the needed improvement to reach the actual concepts of IAs. The following section (section 5) deals with the use of Mobile Agents for Network Management purposes. Section 6 focuses on the architectural aspect of IAs that are used in NM. After presenting the approaches using each agent architecture, the benefits and the weaknesses of the latter are presented according to a NM view. Afterwards, section 7 discusses the cooperation aspect of IAs and how it is handled in the different agent-based approaches to NM. Finally, section 8 presents a unique application of an interface agent for network supervision.

The third and last part (section 9) of this report is a general synthesis where first some important aspects of agency which were not (well) innvestigated are presented, then the new trends in Network Management consideration of Intelligent Agents are discussed.

Finally, section 10 concludes this report.

2 Network Management Issues

Network Management systems have to evolve quickly to satisfy the customer's requirements. Reading the proceedings of international symposium on Network Management, it is clear that there is a strong need to go to flexible and scaleable platforms to fulfill the market expectations. Intelligent Agent paradigm as it was stated in the introduction seems to be a solution to these problems, but if most all Intelligent Agents use networks, only few are used for Network Management. Only two per cent of the official Agent product and research activities referenced by the Agent Organization are related to Network Management. Why?

We see at least two reasons for that:

- Firstly NM is still strongly influenced by OOP, and the huge development effort brought by the companies developing network solutions are always on the way. Most of them focus their strength to use CORBA which is a logical extension of their Object Orientation.
- Secondly, if Agent Oriented Programming has a strong potential in all computer sciences, like the Object technology 10 years ago, it represents a big cut with the classical development techniques, and is not yet mature.

In this context, is it a good idea to spend time studying new development paradigm? To answer this question, first of all let's remind Network Management main objectives and try to emphasize the reasons of the user's unsatisfaction with current Management Systems.

2.1 Network Management Objectives

At first glance and without starting a complex research, as network end-users, it is obvious that we expect fast, secure and reliable connections, as network manager we would like to easily configure and control network access and resources, and as corporate manager we expect a low usage cost. To go further let's see the breakdown of user's network management requirements proposed by [3] and reported in [4]:

- **Controlling Corporate strategic assets**: Networks and distributed computing resources are increasingly vital resources for most organizations. Without effective control, these resources do not provide the pay-back that corporate management requires.
- **Controlling complexity**: The continued growth in the number of network components, end users, interfaces, protocols, and vendors threatens management with loss of control over what is connected to the network and how network resources are used.

- **Improving services**: End users expect the same or improved service as the information and computing resources of the organization grow and distribute.
- **Balancing various needs**: The information and computing resources of an organization must provide a spectrum of end users with various applications at given levels of support, with specific requirements in the areas of performance, availability, and security. The network manager must assign and control resources to balance these various needs.
- **Reducing down-time**: As the network resources of an organization become more important, minimum availability requirement approach 100 percent. In addition to proper redundant design, network management has an indispensable role to play in ensuring high availability of its resources.
- **Controlling costs**: Resource utilization must be monitored and controlled to enable essential end-user needs to be satisfied with reasonable cost.

Because of its more or less natural heterogeneity, networks are difficult and complex to manage. Management systems must handle a wide and increasing number of devices, resources, protocols over larger and larger areas. This is why the Working Group ISO/TC 97/SC 21/WG4 was formed within the ISO in the 1980's [5]. But the work was not easy, and in 1987 IETF taking a more pragmatic approach wrote Network Management proposals for Internet management. Protocols (SNMP) and MIBs have been standardized and quickly implemented on all network devices and became a de facto standard. But unlike OSI, IETF does not provide Management Architecture, which makes these standard difficult and even impossible to use to manage wide networks.

2.2 Network Management and ISO

Network management has been and is always studied by International Standard Organization which defines an architecture and a set of standards. These standards are divided in three main parts:

- Management Communication (CMIS/CMIP) (ISO/IEC 9595...)
- System Management Functions (ISO/IEC 10164-*)
- Management Information (ISO/IEC 10165-*,10589,...)

The five OSI Management Functional Areas The functional side of the OSI Management is split in five strongly linked areas. Functional units are defined and used to cover the functional areas.

- **Configuration Management**: startup, shutdown, setting, reading, modifying configuration
- **Fault Management**: detection, trace, location, analyze, prediction, correction
- **Performance Management**: measurement, data collection, analyze, tuning, testing
- **Security Management**: controlling access, transfer , testing
- **Accounting Management**: set quota, verification, planing, billing

The OSI definea how a "managing system" may communicate and act directly on a managed system which may be either another managing system in charge of a part of the network, or directly an agent.

The management is viewed through the five areas of management (FCAPS), which use "management functions" to perform the management activities. These functions use services over

management protocols (CMISE/CMIP) to access the managed object within the Managed System. The managed system is represented by a management Agent, CMIP Agent, which is in turn in charge of performing on the objects what has been requested by the managed system, or to send event reports when necessary.

In comparison to the Intenet standard of network management which is SNMP, OSI management standard are complex and costly to implement.

2.3 Internet Network Management

The Internet Network Management is a flat management based on protocols of communication between managers and agents to access device Management Information Base (MIB) which identify the variables that can be managed at the agent level. This simple and direct access to devices to be monitored and controlled explains why there is no need for special standards describing the Internet Management Architecture. But the other side of the coin is that SNMP is not usable to manage wide network because of, intensive use of polling which consume lot of bandwidth, problem of packets size to get back MIB data, lack of security, impossibility to have intermediary network manager (manager to manager). Therefore IETF has been obliged to propose improved version of SNMP to solve these problems. The main extensions of the far more complex SNMPv2 are:

- **Protocol**: new primitives GetBulkRequest and Inform Request
- **SMI extensions**: new data types, table categories
- **Manager to Manager capability**: Inform Request, Get Response

The second improvement proposed by IETF is RMON specification developed to perform proactive LAN monitoring on local or remote segments. Both SNMPv2 and RMON2 are a step towards distributed management, but the principle is still to have a centralized management platform. Even if there is no architecture provided, and only low level operations, the Internet NM is far away the most used because of its simplicity.

2.4 Network Management Systems: The Market

All networks are different: topology, media, architectures, hardware, software provide a combinational explosion of network implementation. It's why it's almost impossible to find a evaluation of the Network Management systems available on the market.

Some trials have been done [6] but they are limited and even if the testbed is relatively simple, the results are not very encouraging. Several products are based on the Open View platform completed by ISV (Independent Software Vendor) partner applications that may be executed from the GUI.

In his conclusion I. G. Ghetie [7], who give us a study on the mains marketed NM products, observes that there is a lack of cooperation and integration between Network Management applications even within the same platform.

Conclusion Without going down to details, one may retain the following points:

- None of the products is able to fully cover all the network management areas, they focus on network monitoring and event reporting
- None of the product is able to easily manage heterogeneous networks (lack of useful standard)
- Scaling is difficult due to the inadequacy of the provided application development tools
- The systems are very resource consuming (e.g. SNMP polling)
- All are expensive

Customers are still looking for more simplicity, better integration and collaboration between the different applications, scalable and inexpensive platforms.

2.5 Intelligent Agents and Network Management

Therefore it is definitively necessary to review the way to design Network Management Architecture, with the mind that we will have to use intelligent entities to do the work. Some people propose new ways to develop Network Management Architectures, as Aiko Pra [5] who presents kind of prototyping approach of Network design, or D Stevenson [8] who starts from the "help Desk point of view" to gather the most important NM Requirements and who emphasis that Artificial Intelligence needs to be applied at all management levels.

Studying the great potentiality of the Agents, we feel closer to this last approach, thinking that we have people able to perform activities, or who may acquire skills to perform them, and we will have to organize the NM activities around them. The Network Management Architecture becomes then a problem of organization, where are taken in account the problems encountered by the field services, or help desk.

Several studies are done around what is called the Management Policy, which is based on the Object Oriented paradigm, but prototypes that have been developed are too much influenced by OOP and seem restrictive compared to what may be done with Intelligent Agents. For example the Department of Computing of the Imperial College of London, works on "Authorization-Obligation policies" [9] , and framework for distributed systems management based on Role theory [10]. The interest of these studies is the possibility to provide an architectured approach of Network Management with Intelligent Agents, keeping the simplicity supplied by the Internet management protocols.

3 Intelligent Agent Principles

3.1 Origins

There is no doubt that we are at the time when we may expect software systems to help us, take decisions, work alone, i.e. without being under our permanent supervision. During our exploration, we found a lot of different Agents, Software Agents, Intelligent Agents, Mobile Agents, or Personnel Agents also called Personal Digital Assistant (PDA). At the time where the main software corporations focus their strengths on the ObjectWorld, why is there more and more interest to all these Agents? Reading the many publications and studies, on this subject, we may find three main domains from where this phenomena comes from.

1. Artificial Intelligence, which has been seen for a long time as a marginal software activity, giving complex and difficult software to develop, has been somewhere influenced by the Object Oriented approach [11], and understood the interest to work on simpler, more flexible and re-usable software entities. "The group of agents is more than the sum of the capabilities of its members" say J. Muller into The Design of Intelligent Agents.

2. ObjectWorld after the integration of the network environment with the development of distributed Object Architectures, had a strong need to go a step further providing some "autonomy" and "mobility" to its Objects. New Object oriented languages have been conceived, giving mobility to the objects, and the name agent appears with the following definition "An agent is a computer program that acts autonomously on behalf of a person or organization." [12].

3. Networks, which use since a long time software called "agents" for Network Management purpose, even if its agents have not strong relations with the actual definitions of Intelligent Agents. The SNMP and CMIP agents are software modules gathering and recording management information for one or more network elements and communicating the information on request by a monitor using SNMP or CMIP protocols. There are experiments to replace simple SNMP or CMIP agents by intelligent substitutes [13] but without a global review of Management Principle this should not resolve the current issues. In the other hand the expansion of networks has given a new orientation to the classical way of using computers.

The resources are now distributed among different, and most of the time heterogeneous, systems, and may be shared. In the other direction the information are now spread out on networks, and should be gathered. The standard client/server architecture based on a single server should be reviewed, and the agent approach is now taken in account and is seen as THE solution by companies as Oracle with its client/agent/server architecture.

Even if it's not yet recognized, all these domains should converge, because they are using the same networked environment, and because of economical reason which prevent to design again what already exist.

So far the multiple origins of the Agents make difficult the try to reach a consensus on an Agent definition. As example during the ATAL'96 workshop "Agent Theories, Architectures, and Languages" a lot of discussions were about the definition and Taxonomy of Agents. Inside Mario Tokoro give us a resolute definition as title of his paper "An Agent is an Individual that has Consciousness". But the general conclusion, is perhaps that it exists no urgent need to give an unique Intelligent Agent definition.

This is why in this study we will not join this interesting debate, but on the contrary we will use the Intelligent Agent as a set of capabilities, also called properties, and we will use the word Agent as abbreviation of Intelligent Agent.

3.2 Agent properties

The first aim of this approach by properties has been to make a classification of Agents that we met during our study, but it has been soon clear that this will be not usable as we explain in chapter IV. However it is useful to know what properties Agents may own to understand why they will change our life, and we propose below a description of these properties.

- **Autonomy**

  "Self-government independence: Branch managers have full autonomy in their own areas" [Oxford advanced learner's dictionary]. He decides himself when and under which condition he will perform what actions. "An autonomous agent is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the further" [14]. M. Wooldrige disagrees with Franklin and Graesser's definition of agent, and in "A response to Franklin and Graesser" the autonomy is explicitly required as property, but also the reactive, pro-active and social behavior. One question that we have longingly debated is "what is the autonomy without pro-active or reactive or deliberative behavior?". Someone gives viruses and "networked viruses" also called worms as example of autonomous, and mobile agents.

- **Communication**

  One of the key property of agents, which is the ability to speak with a peer, with human (Interface Agents), or with devices. Communication between agents, called languages often used are:

  - Blackboard: Agents read and write messages in a shared location (blackboard)
  - KQML: Knowledge Query and Manipulation Language is a language and protocol for exchanging information and knowledge using "performatives".
  - KIF: Knowledge Interchange Format
  - COOL: structured conversation, KQML based, used for the coordination of agents [15].

- **Collaboration/Cooperation** Agents are collaboratives when they are able to work together. Also the agent is able to communicate and negotiate with the others, he is deliberative and may coordinate his actions with the others. Collaborative agents are usefull among other things when a task involves several systems on the network. The negotiation is the main issue of the collaborative agents. If coordination may occur without collaboration, collaboration needs negotiation.

- **Deliberation**

  Know rules, and apply them without waiting for instructions. M. Wooldrige and N. Jennings define an deliberative agent as "one that contains an explicitly represented, symbolic model of the world, and in which decisions (...) are made via logical (or at least pseudo-logical) reasoning, based on pattern matching and symbolic manipulation" [16].

- **Mobility**

  Since Java had appeared, we may find a lot of mobile Agents, but there is different kinds of mobility:

  - The mobility which allows the agent to move from one system to one similar one
  - The mobility which allows the agent to move to another different system
  - The mobility which allows the agents to suspend their action on one system, move to another and go on.
  - The mobility which allow the agent to move himself, versus to be transported.
  - The mobility which is a duplication of the agent to another system (cloning).
  - The agent carries his knowledge to another system.

  Generally mobility draws a mix of these definitions. One of the main issue of the mobility is the potential security weakness of mobile agents.

- **Learning**

  There is a least two definitions of learning:

  > The bad one: an agent is said to learn if he is able to acquire knowledge (data).
  >
  > The good one: an agent is said to learn if he is able to use is new knowledge to modify his behavior.

  Despite the fact that learning is an important factor of intelligence, there is only few agents able to learn. Most of the time they have fixed (pre-compiled) rules and knowledge base.

  The objective of learning is for the agent to perform new tasks dynamically without being stopped. Different ways of learning are studied and experimented: Generalization: you observe your environment and deduct rules Instruction: get knowledge and rules from others (transfer)

- **Pro-activeness**

  "Pro-active actions are intended to cause changes, rather than just reacting to change". Pro-active agents generally follow plans, or at least execute rules when the environment reaches a known threshold.

  Sometimes pro-active is used with the same meaning as deliberative, but an agent may be pro-active, because he has been requested to perform pro-active tasks, as opposed to deliberative agents, who decide themselves to be pro-active.

- **Reactivity**

  Do something when an event occurs

- **Security**

  Is able to discriminate friend from enemy and contaminated elements

- **Planning**

  The agent organizes by priorities the actions to perform during is life. For many researchers planning is one of the most important properties for an intelligent agent to possess. Planning is used by deliberative and pro-active agents according to their knowledge of the environment and the possible actions to apply to it.

- **Delegation**

  An agent may ask someone else to perform one of his goals or tasks. This capability his very important to make resource balancing.

Conclusion The above descriptions or definitions, have been subject to interesting internal debates. As it may be seen in the published papers, it is first of all important to have an internal common and well understanding of the terminology used within any projects, for lack of a world-wide consensus.

3.3 Agent Technologies

The development of agents requires languages to build the agents, protocols and languages to give software the possibility to communicate, and languages to describe and transfer agent's Knowledge, Beliefs, Goals, Desires, Intentions [17].

3.3.1 The birth of an Agent

The main idea for our purpose of Network Management, is the possibility for the agent to exist on different heterogeneous environments. Thus they need a standard infrastructure on each system where they need to be hosted. Then agents may be developed like if they will be always on the same machine called the Virtual Machine (VM). The easiest languages to create an "agent" are script languages that may be then executed on remote system, because they are interpreted as shell languages are on Unix systems. The most known are Telescript from General Magic (replaced now by Odyssey which is 100% Java), Perl and Tcl (Tool Command Language) from Sun, and of course Java which is an Object Oriented scripting Language. The second class of languages contains languages coming from classical programming approach as LALO (Agent Oriented Language) which generates C++ sources, or AI oriented languages as Obliq from DEC. Depending the languages and tools used to create them, Agents are born with more or less usable "Intellectual capabilities".

3.3.2 The Agent's Voice

Once the agents are created, they need to communicate with humans or with fellows Depending on the techniques used to creat them, different solutions are available, generally TCP/IP based, communication may have several layers which at the end are used to transfer the information. KQML (see section 3.2) is the most often used language to encapsulate requests and answers between agents over protocols as IIOP, HTTP, SMTP. We may compare here the way we use to communicate ourselves e.g. mail, e-mail, phone, ...

3.3.3 The Agent's knowledge

Now the knowledge transported is described using another layer of language, that we may compare with our respective languages French, English, Portuguese, German, ... One usually used knowledge language is KIF (Knowledge Interchange Format), but specific languages are usually developed within projects for specific purposes (e.g. GDL for MANIA).

It is sometimes not well perceived that Intelligent Agent paradigm use a human view and speech (Anthropomorphism) to present what is only software. This approach is nevertheless normal and accurate, because we are at the stage to use Intelligent Agents as if they were human workers, given them responsibilities and autonomy in their actions. In the next chapters we review the experimentations using agents to solve the problems encountered by the classical Network Management approaches:

- **Scalability**

  When systems are growing or new services are added, Agents may take in charge the new environment, by changing theirs roles, believes, organization.

- **Robustness**

  The fully distributed versus centralized Network Management is more robust because NM operations may continue even if part of the network is disconnected from the Master management site.

- **Upgradability**

  New Agent may replace oldest without stopping their activities.

- **Performances**

  Closer from the systems to manage, the distributed management allow faster reactivity and even pro-activity, and less resources consumption.

4 Intelligent Agents through Management by Delegation

4.1 Preamble

The management by delegation (MbD) paradigm has been developed without initial consideration to intelligent agents. However, we have decided to include MbD in this report for the following reasons.

- Recently, a lot of work that reconsiders MbD is being conducted under the banner of intelligent agents.
- We believe that MbD is one important issue that lead to thinking about intelligent agents in network management.
- Some researchers believe that intelligent agents could be developed by using MbD concepts.

Next, we will respectively present the reasons of introducing MbD, its concepts and principles, the pretended works on "intelligent agents" using these concepts and finally a synthesis and discussion.

4.2 Background

People working in the NM field have quickly identified deficiencies in classical NM architectures and protocols. Management data is distributed among agents with very limited processing capabilities, while data processing to extract management information is centralized in managers. The latter consequently suffer from heavy processing load. Furthermore, management operations must be decomposed to very primitive functions mostly limited to SET- and GET-like primitives and then communicated to the agents. This is referred to be micro-management [18]. Micro-management leads to heavy management traffic in the network.

To overcome this uncomfortable situation, and in order to define a more flexible and scalable architecture with real-time management capabilities, it is tacitly admitted that management functions and operations should be dynamically carried out close to where the managed objects are. Management by delegation is designed following this principle.

4.3 Management by Delegation

The main concept on which MbD relies is that of **elastic server** [19]. An elastic server is an enhanced server whose functionality can be extended and contracted dynamically during runtime. A delegation protocol allows a client to pass new functions to an elastic server and then asks the server to execute these functions by instantiating them. It also allows to keep control on each instance so as it can be stopped then resumed for example. A new functionality is prescribed in a script with no restrictions on the language in which it is written.

Applied to network management, elastic servers can be interpolated between managers and agents (in a similar way to middle level managers in organizations) with similar roles to proxy

agents [4]. These elastic severs are then called MbD agents [18], MAD agents (Manager Agent Delegation) [18], managing agents [20], delegation and flexible agents [21] or elastic agents [22].

An alternative approach would be to make classical SNMP/CMIP agents evolve towards elastic agents. But this approach is rather rejected. Indeed, manufacturers usually tend to keep the legacy systems as they are and to deseign ad hoc components instead.

The manager uses the delegation protocol to upload management scripts to the flexible agents. Using the same protocol, it asks to instantiate some delegated script and to run it. Therefore, the management operation prescribed in this script can execute "autonomously" in the same environment as the managed objects. The manager is able to control its execution via dedicated primitives in the delegation protocol.

Later, Goldszmidt [19] called these scripts "delegated agents". Under the banner of intelligent agents, some researches are being conducted to explore the MbD paradigm for NM purposes. In that, [23] suggests a spreadsheet scripting environment for SNMP. The spreadsheet scripting language allows a manager to prescribe computations that can be carried out by the agent. Each cell in the spreadsheet is defined by an expression that computes a value from other given data such as values in the MIB. Expressions can be inserted, updated and deleted according to the manager needs.

A further improvement is depicted in [24]. Each agent contains functional objects with network management functions allowing to access the management objects and to communicate with other agents. Therefore, a manager needs only to delegate script skeletons invoking these management functions. When running an instance, the management functions' invocation are bounded to those implemented in the functional objects. A same management function could be implemented differently on different agents according to the network device specifics for example.

Other works may use different scripting environments such as the SQL-like one presented in [25], event-driven scripts presented in [26] in its divide and conquer approach and Tcl based scripting language in [27].

Alternatively, Trommer and Knonopka [20] chose to add a rule processing unit to agents. Rules can be transmitted from managers to agents, which can endow the latter with intelligent behavior. Rules can be dynamically loaded, updated and removed, making the agents flexible.

Referring to the agent definition and properties in [16], Mountzia [21] suggested to evolve delegation agents into intelligent agents by making them autonomous, pro-active, reactive, mobile and able to learn. As they are presented in that reference, these ideas are still in their scratch.

4.4 Synthesis and Discussion

The implementations of the Management by Delegation concepts presented above are far from the agency's concept though they are presented under the banner of intelligent agents. Referring to the properties of software agents presented in section 3.2, none of these are really existing in these implementations. The potential properties that are pretended to exist are autonomy, delegation and cooperation. However, delegation agents are not autonomous since management scripts must specify exactly what the agent should do. Agents cannot perform tasks without being ordered to do these tasks. They cannot decide on their own what, when, and how to do. These same critics apply to delegation: only low-level actions can be delegated. Finally, agents are not cooperative, in that they do not dynamically decide to share an important goal in order to achieve it in a coordinated manner.

Nevertheless, the management by delegation concept is tacitly powerful and can be exploited in several ways. It can be the basis to support intelligent agents concepts within two potential manners.

In a first way, one may follow the direction of [21] and conceive delegation agents as software agents instead. The second way consists in delegating intelligent agents as management scripts. This leads to remote-execution mobile agents. In both cases, further efforts, improvements and considerations must be addressed. Actually, an agent-oriented view of delegation will shift from the delegation of scripts and functions to the delegation of goals which is a high-level delegation.

5 Network Management with Mobile Agents

5.1 Generalities

Mobile agents (MA) are able to hop from node to node during their activities. Therefore, they can use the information they found during their past visit to sites in order to adjust their behaviour accordingly. They are also able to carry out activities close to where the needed resources are. In the following we present how these properties could be used in Network Management and then discuss the presented works.

5.2 Advantageous ways of using MAs

In [28], Thomas Magedanz has presented issues for the deployment of MAs for NM activities purposes. MAs can encapsulate management scripts and be dispatched on demand where needed. An agent can be sent to a network domain and travels among its elements collecting management data, and return back with the data filtered and processed. Sending a MA for this task is a substitute to performing low-level monitoring operations and process them centrally. Then if the agent is able to extract useful and concise information from raw data collected on each element, then the agent's size can remain small enough to save bandwidth usage. Actually, this is one of the most advanced arguments to promote MAs.

Similarly, a MA can also be used to achieve management operations. Instead of remotely invoking management operations, the administrator can encapsulate these operations into a mobile agent and send it where needed. If these operations involve a lot of message exchange along with human decision making, then the mobile agent can also save bandwidth by locally taking the necessary decisions while performing those operations.

These ideas have been implemented by FTP Software [29]. The IP Auditor is an application that dispatches mobile agents in the target network in order to collect management information such as configurations and network elements' operation states. The IP Distributor is another application that dispatches agents that distribute management payload.

Another original and concrete application of MAs is presented in [30]. Mobile agents are used to control traffic congestion in a circuit-switched network. A first class of MAs, called *parent agents*, randomly navigate among the network nodes and collect information about their utilization. By keeping track of these information, they have an approximated utilization average of the network nodes. Therefore, they are able to identify which nodes are congestioned relatively to this average. When a congestioned node is identified, a load-balancing MA is created. It has to update the routing tables of the neighboring nodes (using an optimal algorithm) so as to reduce the traffic routed by the congestioned node.

In a comprehensive study on code mobility, [31] presents a consequent advantage of using mobile agents; When the network administrator can only be connected via an unreliable, costly or lossy link, he can create its MA off-line, connect to the network to dispatch the agent, close the connection and then reconnect later to get his agent back with the results. This principle is actually implemented in Astrolog [32] to support mobile managers; In order to have a network flexibly managed, an administrator can connect to the network using its portable computer. The connection could even be established on the basis of a GSM link. If a hand-over occurs, the dispatched MA can retrieve its sender's new location and return back to him.

[28] and [33] suggest other possibilities of applying MAs. A service or a network provider can send MAs to user end-points in order to adapt his equipment to new services. These agents can also achieve other tasks such as user accounting and capturing user requirements. [34] also suggests to use MAs for immediate service creation and customization.

5.3 Discussion

The use of MAs seems to be rosy for NM Applications. However, some precautions must be taken before heavily investigating in this direction. Concerning bandwidth saving, it is not obvious

that carrying an agent with high capabilities consumes less bandwidth than carrying the desired information and processing it centrally. Actually, management information processing (e.g for event correlation or fault source detection) is often complex to achieve. Unless the agent is described in a high-level and concise language, it will be of a large volume. This leads to two options. Either the Agent run-time environment would be complex and heavy, or the agent has only limited capabilities. The first option leads to costly implementation of a MA platform while the second may simply be useless. Yet the design of a run-time environment for mobility which supplies the agents with a homogeneous interface across heterogeneous systems is not straightforward to conceive. The Java virtual machine is already a forward step but this is still far from the agent's concept.

Finally, security is one of the most challenging problems that MAs may face. In a sensitive domain such as NM, security must be carefully studied. [29] has already claimed that the IP Auditor's and IP Distributor's agents (see the previous sub-section) support security mechanisms.

Therefore, MAs, though presenting a seducing approach, must be carefully investigated and deployed. In particular, using MAs for predetermined tasks such as in [30] is needless though elegant. Instead, a protocol for exchanging utilization information and traffic routing could have been used substantially.

6 Reactive, Deliberative and Hybrid Agents in Network Management

6.1 Preamble

According to its architecture, an agent might be reactive, deliberative or hybrid [16]. While a reactive agent acts in situation-reaction or event-reaction manner, a deliberative agent acts according to decisions taken after a reasoning process over the environment. A hybrid agent is a combination of both. The following three sections will respectively treat these types of agents, and the last section will provide a synthesis and a discussion.

6.2 Reactive Agents

The application presented in section 5 used Mobile Agents for congestion management. The agents used therein were reactive agents [30]. Both the parent agents and the load agents do not use any model of the network they are managing. Instead, they only observe the state of the network nodes and react accordingly. The use of Reactive Agents is particularly advantageous for this application because MAs need to have a light weight not to overload the network by their mobility. The parent agent do not react unless they discover an overloaded node. The decision to react by creating a load agent is taken by simply comparing the currently visited node utilization with the average utilization computed during its traveling. The load agents which update the routing tables decide on the basis of an algorithm which doesn't imply any reasoning. Each agent taken alone, be it a parent or a load agent, would not be able to manage the network. The application is therefore a good example that shows how the overall multi-agent system goes beyond the capabilities of all the reactive agents when taken individually.

Other researches concentrate on building agent-based frameworks for network management and use the reactive architecture mainly for its simplicity. In [35], intelligent agents are applied to implement distributed management policies. Within this framework, agents are composed of a communication mechanism, a rule base, a solution base and an inference engine. The rule base contains rules that matches perceived events (corresponding to anomalies) with the reactions to be taken. Reactions (called services within the paper) are also described by rules within the solution base. Therefore, the reactive rules are of the form `<event-Id,reaction-Id>`. This framework was used to achieve usage-based accounting.

Another framework has been described in [36]. A generic agent architecture is defined and built using April++, an Object-Oriented extension of April [37]. The agent is organized in units which

communicate (*inside the agent*) via a blackboard mechanism. The units are organized vertically which means that they execute concurrently. The local information of the agent is centralized in an information database. Besides, the agent has a link mediator which is an interface with the physical components of the network, and a head that insures the communication with other agents. New functions can be added to the agent by acquiring new units. This model has been applied to provide agents suitable for multi-service network management. Indeed, for this to be achieved, the agent is endowed with a customer unit to interact with the user, a service unit that honors service requests and a fault handling unit that deals with the faults that may occur.

6.3 Deliberative Agents

Two applications can be stated as deploying purely deliberative agents. The first is that described in [38]. The paper describes an interesting framework applied to control a VPN service. Here agents are used to automate the negotiation that used to occur between the service provider and the service users when these ask to change to change the service parameters or to repair network faults. A customer agent has knowledge about the logical structure of the VPN and its usage, while the provider agent knows both the logical and physical implementation of each customer's VPN. Thus, agents have a model of the external world on which they act. Precisely, a customer agent knows for each trunk (or logical link) its capacity and its utilization, whereas the provider agent knows for each trunk the physical links on which it is implemented. When, let's say, a network fault occurs, the VPNs based on the faulty network element are affected. In this case, the customer agent first asks the provider agent to repair the fault. In many cases, a complete and immediate repair is not possible and a negotiation-based cooperation is started. Customer agents try to find intermediate solutions based on their knowledge on the utilization of their respective trunks, and suggest partial solutions to the provider agent by updating the logical structure of their VPNs. The provider agent coordinates these solutions and makes the necessary updates in order to reach an accepted configuration. Both customer agents and provider agents use logical reasoning based on their models of the network and on their beliefs on its elements. Actually, the implementation of this reasoning is based on a planner called PRODIGY.

The second application of deliberative agents in NM is the MANIA project (Managing Awareness in Networks with Intelligent Agents) [39]. An agent-based approach is used to manage the quality of service in the network. Agents are structured in beliefs, desires, intentions, goals and commitments. An agent's beliefs express its perception of the environment. A first part of the beliefs describe a real-time state of the network, e.g. the printing service is highly solicited at the current time. A second part contain the historical behavior of the network. The historical behavior is used to achieve some kind of learning about the dynamics of the network such as deducing that the printing service is highly solicited everyday from 10 to 11 am. A third part of the beliefs translate the states of the services provided over the network, e.g. the minimal response time the NFS server can ever have, or the maximum client number a web server can handle at a time. Finally, the agent may have beliefs on the user application contexts i.e user requirements in terms of QoS.

The agent's desires consist of two parts. The first part of desires correspond to the requests that the agent could not satisfy, such as when a certain user requires a video connection while there is no available bandwidth (according to the agent's belief). The second part consists of motivation policies. The network administrator may want to "motivate" the agent to give a certain priority to some project members because they have a constraining deadline.

When receiving a user-application context (which describes the user needed resources and the required quality of service parameters), the agent translates it into a set of goals. Goals are independent from the system's state. For example, a goal may state to "actively monitor the NFS server". These abstract goals are then mapped into intentions. Intentions take into account the current state of the system, the available means to achieve the goals (e.g. MIBs, testers, etc.) and the possible constraints that may apply.

Finally, and as the administrator could motivate the agent, it is also able to specify obligation policies which form the commitment part of the agent.

6.4 Hybrid Agents

Here also two applications are worth presenting. The first is presented in [40]. The overall project addresses the application of intelligent multi-agent systems to manage connection admission control in ATM. An ATM network is very dynamic and managing it centrally will not be appropriate because managed data get quickly out of date. But local management is not appropriate though because it lacks an overall view of the network. Here, the hybrid agents are particularly interesting. They may combine both a local real-time management via the reactive behavior, while the deliberative behavior is concerned with cooperating with other agents in order to achieve global planning and coordinate high-level tasks.

But since the project is still in its beginning, no further implementation details are given.

In contrast, the second application is already well designed and the agent's architecture is well defined and is based on the concept of **Vivid agents**. Vivid agents [41] comprise both a reactive and a deliberative parts. However, the reactive part of the agent is not hardwired within the agent. Instead, the deliberative part may dynamically change the reactive behavior. **Reagents** are particular vivid agents with no planning capabilities.

Reagents are applied for performing distributed diagnosis on distributed systems [42]. The network is partitioned in physical domains, each domain has its own diagnostic agent. The agent has a detailed knowledge model of its domain and minimal information about the neighboring domains, mainly the address of their respective agents. The multi-agent system applied a distributed version of the Model Based Diagnosis. The principle is that when an agent detects a fault, a lost connection for example, it starts by performing local diagnosis of its domain. This is the deliberative behavior of the agent, since it performs reasoning according to a model of its domain. If it finds no local fault, then it sends the resulting observations to the neighbor agent. The latter performs the same procedure. If it finds the faulty element(s), it reports it to first agent. If the first agent itself had received the observation from another agent, then it must get it informed by forwarding the report. This is a reactive behavior of the agent where no reasoning is performed.

6.5 Synthesis and Discussion

The first thing that's worth noting is that few papers supply the reader with information about the agent architecture. This is attributed to several possible reasons. The first reason is that many papers only present scenarios and do not yet reach the point of designing multi-agent systems. An example of such papers are [28] and [33], [43] (though the reader may implicitly understand that agents may have goals) and [44]. The second reason is that some frameworks support both architectures, and that implementation is not yet started. An example of these is [45]. Finally, this could be due to ignorance of agents' architectures which is rather common for example in the mobile-agent community.

What is interesting to note, is that reactive agents are suitable for local and real-time management operations. They can be used efficiently in fixing well known faults within a relatively small portion of a network. An approach based on relating symptoms to faults, and faults to fixing methods is broadly adequate for this purpose.

On the other hand, deliberative agents are much stronger from problem-solving and logical reasoning point of view than reactive agents. This makes them require much more resources, and makes their response time rather long. For this reason, they are suitable for complex but not time-constrained tasks. A good example would be that of analyzing the origin of a security failure in a large corporate network.

Further, the idea of [40] of using hybrid agent to take the advantages of both architectures is very interesting. The concept of vivid agents is a good basis to implement this idea. Vivid agents offer an increased flexibility when designing an agent since their reactive behavior could be updated during run-time. This could be applied for example to support complex learning algorithms where the agent, still having timely reactions, has its behavior improved in time. However, one may keep in mind that the same benefits could be obtained using a heterogeneous architecture, i.e. a multi-

agent system with both reactive and deliberative agents with the latter governing the behavior of the former (See [2] for details and further references).

7 Cooperation in Multi-Agent Systems for Network Management

Cooperation is one of the most important properties in MASs [46]. In NM, cooperation could have large benefits in Distributed Network Management Systems. Though the term "cooperation" (or "collaboration") is unclearly used, we will try in the following subsections to describe cooperation in agent-based network management approaches progressively starting from no or simple coordination towards more elaborated cooperation.

7.1 Coordination without Communication

As described in section 5, the MAS used in [30] is based on agents which do not directly communicate. An agent may be aware of the other agents only by observing the system's state. Each time a parent agent visits a node, it leaves therein some information such as its identity, its age and the date of its visit. By reading this information in every visited node, the parent agents are aware of their number in the network. If their number is excessive, then the youngest parent agent will terminate. Furthermore, in determined nodes, static processes continuously check if all the parent agents are still alive and did not crash. If a very long period has elapsed from the last visit of a certain parent agent, then the static process will conclude that this parent has crashed and will replace it.

Besides, when a load agent is launched on a node, it creates a record with its identity and its start time. When it finishes updating the routing tables, it returns back to the first node and registers its same start time again and then terminates. Therefore, any parent agent that visits that node is able to know if the load agent is still active or not. It is also able to detect that the load agent has crashed if the start time has not been written for the second time after a long period. The parent agent becomes responsible of creating another load agent to replace the first.

When there is no direct communication between agents, a lot of coherence problems are avoided. Firstly, an agent would not send messages to another agent that has been crashed. Besides, all the agents equally share the same information and no mechanisms are needed to preserve coherence. Secondly, the agents are able to coordinate their tasks in a simple way. These properties gave the MAS thus designed a high degree of robustness and graceful degradation which are rare (actually never) to be considered in the other applications.

7.2 A Primitive Cooperation

In section 6, we presented an agent-based system implementing a distributed version of Model-Base Diagnosis [42]. In case of a fault, an agent performs local diagnosis and if it finds no faulty element, it asks the next agent to perform diagnosis in its corresponding domain. One may consider this as a simple cooperation for at least two reasons. The first reason is that the underlying communication mechanism is itself primitive. The second is that this cooperation is hard coded, and the agents have not decided how to carry it out. However, and referring to [46], agents are cooperating since they participate to achieve a global (though implicit) goal which is to identify the faulty element in a large network.

7.3 Self-interested Negotiation Agents

Self-interested agents do not need to know about the other agents' plans and goals. Instead, they work in order to achieve their own goals in the best way without caring about the others. However, in order to achieve a goal, a self-interested agent may need the services of some other agents. A natural approach would be based upon negotiation. Here, when requiring some task to be done

by another, the agent tries to maximize its profit by launching a kind of auction. A good model for this is the economic market place [47].

Self-interested negotiation agents seem to be very suitable for some network management activities, especially for service management. Most of the papers dealing with service management refer to this kind of agents. In [28], Thomas Magedanz suggests that these agents are adequate for the provision of high-level services in an open electronic market of telecommunications services. A good basis to cover this trend is the FIPA application draft presented in [44]. The application defines three kinds of agents, namely the Personal Agent (PA), the Service Provider Agent (SPA), and the Network Provider Agent (NPA). The latter is responsible for the provision of the network resources and elements which are necessary for the service implementation. Based on these network resources, the SPA is responsible for the provision of the services with the expected quality. The PA is a kind of Personal Digital Assistant [48] that assists the user in defining his requirements for the application's needs with regard to the user's preferences. It then has to negotiate these requirements with different SPAs in order to find out the best provider with the best service in terms of maximum quality and minimum cost. When a communication is about to be established, the PA has to commit the local resources of the user and to configure his equipment according to the established connection.

The SPA has to catch the user's requirements and to identify the necessary services and to map them into these services' parameters. It then negotiate with the NPAs to select the best network provision, again in terms of maximum quality with minimum cost. Feedback with the PA may occur in order to conclude with the best arrangement both with the Network Provider and the User.

Finally, the NPA gets the SPA's specifications and translate them into network requirements in terms of bandwidth, jitter, etc. It may also have to negotiate with other NPAs, mostly in the case where multiple network domains are implied.

Other works on the subject use the same, otherwise similar (restricted) approaches. Mike Ruzzo and Ian A. Utting [43] have also introduced User Agents to allow for the definition of customized services. New services can be customized by supplying the agent with the user's policy. The UA then is able to make the best choice between what can be achieved as suggested by the fall-back of the called agent, and what the service provider offers. The caller and the called agents may also negotiate in order to reach an agreement that satisfies both users' policies i.e. preferences and constraints. To support user mobility, end-point agents are also introduced. Mobile-user agents have to negotiate with end-user agents to get the permission to access the corresponding end-points and establish communications.

Other scenarios suggest to implement new services in agents. [HJ97] suggests that while services are user-customized by PAs, new services can be created within service agents in a service-level agent environment. Service agents encapsulate network specifics and each one is dedicated to one service that can be constructed over other service agents. Similarly, [MRK96] presents a scenario where mobile agents are sent into an agent environment on the services' host, where they can offer new services by using the legacy services and/or by exploiting other mobile agents' facilities. In both scenarios, the agents that implement new services have to negotiate with the already existing agents in order to find the agents on which the new service is best implemented.

In [40], auction-based negotiation is suggested to match resource providers to consumers in ATM networks. This is a simple yet efficient protocol by which decisions are taken in a timely fashion by "matching highest bidding buyers to lowest bidding sellers".

Negotiation is also adopted in [45] to manage an ATM. Agents are organized within a hierarchy of authorities. Each authority is responsible of some delegated resources and exports "*performance indices*". Therefore, when an authority (particularly, a service agent in the authority) receives a user request, it commits the necessary directly managed resources and then decides which of the sub-authorities is best eligible to route the request through. This decision is based upon a tendering process using the above-mentioned performance indices.

7.4 Delegation-based Cooperation

As opposed to the MbD paradigm presented in section 4 which only deals with script (or at the best functionality) delegation, we address here the delegation between intelligent agents. This is a higher-level delegation because agents delegate goals and motivations instead of low-level operations.

The agent framework described in [35] (see section 6.2) makes use of policies expressed in logical rules. High-level policies are delegated to agents. These policies are then mapped into lower-level policy rules and stored in the agent's rule base.

The MANIA project (see section 6.3) also makes use of high-level delegation of goals. An agent may fail in achieving a goal mainly because it lacks the necessary resources. It, therefore, delegates that goal to another agent. What is important to note is that only the goal and not the associated intentions is delegated. The latter agent rebuilds the entire intentions for that goal and these intentions may differ from those built by the delegating agent due to different beliefs, commitments and skills of the two agents.

7.5 Synthesis and Discussion

Although it is tacitly admitted that cooperation could be very beneficial to distributed network management architectures, it is not yet well exploited in current agent-based approaches. The non-communicative approach presented in [30] has provided robustness and made easy to reach the graceful degradation of the whole MAS. Negotiation is a very interesting-to-use technique for the provision and management of telecommunication services. A market place model is suitable to implement agents that work to provide high-level services with the best quality and the lower price. For this, an agent has to negotiate with others in order to maximize its owner's profit.

The underlying communication mechanism is not equally addressed for each paper. While in [42] the communication protocol is not emphasized (simply because there is no need for more than simple message exchange), [43] has already distinguished between message syntax and semantics. In [38], two types of messages, namely INFORM and REQUEST messages were sufficient, whereas in [36], messages are identified using message patterns. Most of the other related works has opted to KQML like in [44] and [35]. In the MANIA project [39], agents use KQML for the message syntax and a KIF-like Goal Definition Language to exchange management goals. Finally, in [45], an extended version of KQML is used. The added performatives are PROPOSE, COUNTER-PROPOSE, ACCEPT, REJECT, PROBLEM and REJECT. They are introduced to support the negotiation and the tendering processes.

From the Distributed Problem Solving point of view, none of all the presented cooperative agents has reached the degree of elaborating global plannings and collaborate to achieve them. Also, according to [46], a cooperative MAS may either emerge from agents which give priority to their own internal goals or from agents that try first to better achieve the global goals of the system. Most of the cooperative agents described above fall in the first category.

8 An Interface Agent

An application of an interface agent for network supervision is presented in [49]. The agent has to process a large amount of alarms and events by filtering them and relating each notification to its context.

An on-line knowledge acquisition is adopted. Using the chronicle model (see [50]), the agent is able to perform temporal reasoning on the notified events and to automate management tasks. A chronicle is a set of (correlated) events terminated with an action that must be taken when the chronicle is matched. A chronicle is expressed using two formula types. The `hold` formula expresses that an attribute holds its value over some interval. The `event` formula expresses a discrete change of an attribute's value.

While "*looking over the shoulder*" of the human administrator, a **Chronicle Recognition System** identifies chronicles and associates each one to the taken action. A **Learning system**

evaluates the identified chronicle by comparing it to the known ones. A first step would be to store the chronicle in an unconfirmed chronicle base. After reaching a certain threshold (or being confirmed by the human), the chronicle is transfered in a confirmed chronicle base.

In this application, the agent learns from the user either by observing him or by receiving explicit instructions. Cooperating with other interface agents is not yet supported but it is planned for future study (see the conclusion of [49]).

9 A General Review

This section discusses the above presented achievements with an agent-oriented view. It states the agent properties that were not investigated, thus requiring more study to be exploited in the network management field. Afterwards, the trends in the introduction of agent technology by network management community is presented.

9.1 An agent-oriented view

9.1.1 Pro-activeness

As stated in section 3.2, pro-activeness is the ability for an agent to anticipate changes in the environment and act in a way to prevent the potential forthcoming problems that might occur. In the NM context, pro-activeness makes an agent able to foresee faults or QoS performance degradation and to avoid such situations before they even occur.

Pro-activeness is the most ignored property in existent agent-based approaches to NM. Only two approaches have referenced it.

The first is [35] which states when talking about pro-activeness: "*Policies could be used to allow the IA to divide a problem up into sub-problems, and then to solve each one of the sub-problems ...*". Clearly, this statement has nothing to do with pro-activeness.

The second approach is that used in the MANIA project [51]. The pro-active behavior is achieved in two possible ways. In the first way, the agent observes the application contexts sent by the users to declare their QoS requirements. It then deduces what network resources would soon be highly solicited. The second way consists in learning the evolution of the network's behavior during time. The agent could for example deduce that the NFS server is overloaded each day from 9 to 10am. The agent may decide to launch another server just before 9am or to forbid any logging-in on the server's machine during the rush period.

9.1.2 Learning

Learning Network Management knowledge or expertise should be distinguished from learning the network's behavior. The first kind of learning enables the agent to acquire new capabilities in management, mainly to handle new types of network problems. The unique example of this is the interface agent presented in [49] and reported in section 8.

The second kind of learning enables the agent to acquire knowledge on the network it manages in order to apply its very same expertise but more efficiently. In the previous section, we saw how this kind of learning was applied to provide pro-activeness. In [30], one may consider that the agents keep learning the system's state in order to deduce a sufficiently accurate node utilization average.

9.1.3 Robustness and Graceful Degradation

These two properties concern the whole multi-agent system. The graceful degradation means that the MAS does not fail drastically at the boundaries [2]. Robustness is the ability to recover from faults that may occur due to the underlying environment (network failure or system halt).

The unique application where these both properties were studied is that in [30]. The mechanism that detects crashed agents and replace them was already presented in section 7.1. Let's just recall

that if a parent agent crashes, it will be detected and replaced by static processes whose role is to maintain a minimum number of agents. Meanwhile, the other agents continue their tasks without stopping the system (graceful degradation). If a load agent crashes, it will be detected and replaced by the parent agents. Therefore, the MAS keeps on performing its role and is able to recover from possible crashes (robustness).

9.2 Network Management trends

The number of network management firms and network-device manufacturers which are getting interested in Intelligent Agent technology is rapidly increasing. Many manufacturers are trying to incorporate IAs within network devices such as routers and ATM switches. The idea behind is to endow the network devices with a level of intelligence that allows them to acquire the properties of autonomy and self-healing. Besides, this intelligence at the lowest level of the network will have benefits on management traffic reduction and network management automation.

Other trends are focusing on the service level. These trends are mostly lead by telecommunications operators which are addled for a high degree of flexibility in establishing new services and for means to control and satisfy the quality of service that the end user requires.

Network management platforms are also affected by IAs. Many manufacturers are announcing products supporting IAs, although it is not yet specified how IAs will be integrated in their platforms. For example, the whole NM platform can be based on IAs, or it can include a framework to allow network administrators to develop their own NM intelligent agents, or simply can use static preprogrammed intelligent agents. Their hope is for the NMS to be much more reactive (or even pro-active), flexible and scalable.

In fact, a lot of researches are still needed to reach an acceptable usage of IA technology in NM platforms. Indeed, a huge number of languages, environments and techniques are available to design agents and to define their behavior and how they can cooperate. Besides, there is still a lot of yearning in the NM field in order to find solutions to integrate heterogeneous and vendor-specific equipments.

Regardless of these issues, efforts are directed towards Java-based solutions. People are relying on the promise of a portable code due to the Java virtual machine and to Java-enabled chips which are announced and even recently commercialized.

10 Conclusion

This paper has presented a state of the art on Network Management approaches based on Intelligent Agents. The reader should be aware that many researches do not offer any publication due to (industrial) confidentiality constraints. We are informed of such projects only via personal contacts, projects' home pages and from some individual participations of members of such projects to mailing lists.

The reported papers that we could collect are of three classes. The first class of papers only describe scenarios where some NM activities could be achieved by using agents. These papers do not offer any architectural details which makes them difficult to evaluate. The second class of papers present an agent-based solution to a specific NM issue. These solutions are well detailed but could not be used for an integrated approach to NM. The last class of papers describe general architectures based on agents, each with an example of application to a specific management function. Actually, the most addressed functional areas are fault, configuration and performance management. Accounting and security management are much less considered, the latter being never studied.

The use of IAs has provided the presented NM approches with a high degree of automation and flexibility. The human administrator is shifted out from low-level and routine tasks. He becomes able to interact with the agents in a high-level manner using policies, motivations and goals. Another new interesting feature is that agents are closely aware of the detailed state of the

network elements and their usage. Corrective operations can be appropriately taken in a timely fashion and pro-active actions could be much more easily planned.

For these reasons, many companies have already begun heavily investing in IAs, some are even participating to standradization efforts as a way to promote their own researches in this domain.

Credits

This report is established within the framework of the project F&E 391 which is a cooperation between SwissCom and Eurecom Institute. It is financially supported by SwissCom, and any further information about the project might be required from our correspondant M. André PRIM (e-mail: Andre.Prim@SWISSCOM.COM). The authors would like to thank Raul Oliveira for his precious collaboration.

References

[1] Christian Huitema. *Et Dieu créa l'Internet.* Eyrolles, 1996.

[2] Hyacinth S. Nwana. Software agents: An overview. *Knowledge Engineering Review*, 11(3):205–244, October/November 1996. http://www.cs.umbc.edu/agents/introduction/ao/.

[3] K. Terplan. *Communication Networks Management.* Prentice-Hall, 1992.

[4] William Stallings. *SNMP, SNMPv2 and RMON, Practical Network Management.* Addison-Wesley, USA, 1996.

[5] Aiko Pras. *network management architecture.* PhD thesis, Universiteit Twente, Department of Computer Science, February 1995. available from http://wwwsnmp.cs.utwente.nl/~pras/thesis.html.

[6] Christine Gressley. Network management resources - reviews of network management systems, July 1996. available at http://tampico.cso.uiuc.edu/~gressley/netmgmt/reviews/.

[7] I. G. Ghetie. *Networks and Systems Management. Platforms Analysis and Evaluation.* Kluwer Academie Publishers, 1997.

[8] Douglas W. Stevenson. Network management: What it is, what it isn't., 1995. avaialable at http://netman.cit.buffalo.edu/Doc/DStevenson.

[9] Damian Marriot and Morris Sloman. Implementation of a management agent for interpreting obligation policies. In *IEEE/IFIP 7th International Workshop on Distributed Systems and Operations Management - DSOM'96*, L'Aquila, Italy, October 1996.

[10] M. Sloman E.C. Lupu. Towards a role-based framework for distributed systems management. *Journal of Network and Systems Management*, 5(1), 1997.

[11] S. Labidi and W. Lejouad. De l'intelligence artificielle distribuée aux systemes multi-agents. *Rapport de Recherche*, 1(2004), 1993. available at INRIA Sophia-Antipolis 2004 rte des Lucioles, BP 93, 06902 Sophia-Antipolis Cedex (France).

[12] OMG. Mobile agent facility specification. Technical report, Crystaliz, Inc., General Magic, Inc., GMD FOKUS, International Business Machine Corporation, June 1997. OMG TC Document cf/xx-x-xx.

[13] Nick Parkyn. Architecture for intelligent (smart) agents, Jun 1995. available at http://www.citr.uq.oz.au/.

[14] Stan Franklin and Art Graesser. Is it an agent, or just a program?: A taxonomy for autonomous agents. In *Intelligent Agents III. Agent Theories, Architectures, and Languages*. Springer, 1996.

[15] Mihai Barbuceanu and Mark S. Fox. The design of a coordination language for multi-agent systems. In *Intelligent Agents III. Agent Theories, Architectures, and Languages*, pages 341–355. Springer, 1996.

[16] Michael Wooldridge and Nicholas R. Jennings. Intelligent Agents: Theory and Practice. *Knowledge Engineering Review*, 10(2):115–152, 1995.

[17] Jörg P. Muller. *The Design of Intelligent Agents - A Layered Approach*. LNAI State-of-the-Art Survey. Springer, Berlin, Germany, 1996.

[18] Yechiam Yemini, Germán Goldszmidt, and Shaula Yemini. Network Management by Delegation. In *The Second International Symposium on Integrated Network Management*, pages 95–107, Washington, DC, April 1991.

[19] Germán Goldszmidt. Distributed system management via elastic servers. In *IEEE First International Workshop on Systems Management*, pages 31–35, Los Angeles, California, April 1993.

[20] Markus Trommer and Robert Konopka. Distributed network management with dynamic rule-based managing agents. In *Integrated Network Management V : integrated management in a virtual world*, pages 730–741, San Diego, California, USA, May 1997. IFIP, Chapman & Hall.

[21] Maria-Athina Mountzia. Intelligent agents in integrated network and systems management, 1996.

[22] Germán Goldszmidt and Yechiam Yemini. Distributed Management by Delegation. In *The 15th International Conference on Distributed Computing Systems*. IEEE Computer Society, June 1995.

[23] P. Kalyanasundaram, A.S. Sethi, C. Sherwin, and D. Zhu. A spreadsheet-based scripting environment for snmp. In *Fifth IFIP/IEEE International Symposium on Integrated Network Management IM'97*, volume 5, pages 752–765, San-Diego, California, USA, May 1997. Chapman & Hall.

[24] Motohiro Suzuki, Yoshiaki Kiriha, and Shoichiro Nakai. Delegation agents: Design and implementation. In *Integrated Network Management V : integrated management in a virtual world*, volume 5, pages 742–751, San Diego, California, USA, May 1997. IFIP, Chapman & Hall.

[25] Simon Znaty, Michel Lion, and Jean-Pierre Hubaux. Deal: A delegated agent language for developing network management functions. In *First International Conference and Exhibition on the Practical Application of Intelligent Agents and Multi-Agent Technology*, 1996.

[26] R. Kooijman. Divide and conquer in network management using event-driven network area agents. *?*, May 1995. available at http://netman.cit.buffalo.edu/Doc/Papers/koo9505.ps.

[27] Garry Grimes and Brian P. Alley. Intelligent agents for network fault diagnosis and testing. In *Integrated Network Management V : integrated management in a virtual world*, pages 232–244, San Diego, California, USA, May 1997. IFIP, Chapman & Hall.

[28] Thomas Magedanz. On the impacts of intelligent agents concepts on future telecommunication environments. In *Third International Conference on Intelligence in Broadband Services and Networks*, Crete, Greece, October 1995.

[29] FTP Software. Ftp software agent technology. Technical report, FTP Software, http://www.ftp.com/product/whitepapers/4agent.htm, 1997.

[30] S. Appleby and S. Steward. Software agents for control. In P. Cochrane and PJT Mealthey, editors, *Modelling Future Telecommunication Systems*. Chapman & Hall, 1994.

[31] Mario Baldi, Silvano Gai, and Gian Piertro Picoo. Exploiting code mobility in decentralized and flexible network management. In *Proceedings of the First International Workshop on Mobile Agents*, Berlin, Germany, April 1997. available at http://www.polito.it/~picco/papers/ma97.ps.gz.

[32] Akhil Sahai, Stéphane Billiart, and Christine Morin. Astrolog: A distributed and dynamic environment for network and system management. In *Proceedings of the 1st European Information Infrastructure User Conference*, Germany, February 1997. Available at http://www.irisa.fr/solidor/doc/pub97.html.

[33] T. Magedanz, K. Rothermel, and S. Krause. Intelligent agents: An emerging technology for next generation telecommunications? In *INFOCOM' 96*, pages 464–472, USA, March 24-28 1996. IEEE.

[34] Gisli Hjálmtýsson and A. Jain. An agent-based approach to service management - towards service independent network architecture. In *Integrated Network Management V : integrated management in a virtual world*, pages 715–729, San Diego, California, USA, May 1997. IFIP, Chapman & Hall.

[35] P. Steenekamp and J. Roos. Implementation of distributed systems management policies: A framework for the application of intelligent agent technology. In *2nd International Workshop on Systems Management*, Toronto, Ontario, Canada, June 1996. IEEE.

[36] Nikolaos Skarmaeas and Keith L. Clark. Process oriented programming for agent-based network management. In *ECAI96 Workshop on Intelligent Agents for Telecommunication Applications (IATA96)*, Budapest, Hungary, August 12 - 16 1996.

[37] F. G. McCabe and K. L. Clark. April: Agent process interaction language. Technical report, Dept. of Computing, London, UK, November 1994. available from http://www-lp.doc.ic.ac.uk/~klc/.

[38] Robert Weihmayer and Ming Tan. Modelling cooperative agents for customer network control using planning and agent-oriented programming, 1992.

[39] Raul Oliveira and Jacques Labetoulle. Intelligent agents : a way to reduce the gap between applications and networks. In J. D. Decotignie, editor, *Proceedings of the First IEEE International Workshop on Factory Communications Systems - WFCS'95*, pages 81–90, Leysin, Switzerland, October 4-6 1995. Available at http://www.eurecom.fr/~oliveira/wfcs/wfcs.ps.gz.

[40] M. A. Gibney and N. R. Jennings. Market based multi-agent systems for atm network management. In *Proc. 4th Communications Networks Symposium*, Manchester, UK, 1997. Available from http://www.elec.qmw.ac.uk/dai/projects/agentCAC/.

[41] Gerd Wagner. Vivid agents - how they deliberate, how they react, how they are verified. In W. Van de Velde and J.W. Perram, editors, *Agents Breaking Away, Proc. of MAAMAW'96*, 1996. Available from http://www.informatik.uni-leipzig.de/~gwagner/.

[42] Peter Fröhlich, Iara de Almeida Móra, Wolfgang Nejdl, and Michael Schroeder. Diagnostic agents for distributed systems. In *Proceedings of ModelAge97*, Sienna, Italy, Junary 1997. Available at http://www.kbs.uni-hannover.de/paper/96/ma1.ps.

[43] Mike Rizzo and Ian A. Utting. An agent-based model for the provision of advanced telecommunications services. *?*, 1995.

[44] Foundation for Intelligent Physical Agents. Application design test: Network management and provisioning. http://drogo.cselt.it/fipa/spec, June 1997.

[45] Fergal Somers. Hybrid: Unifying centralised and distributed management for large high-speed networks. In *Networks Operation and Maintenance Symposium (NOMS96)*, Kyoto, 1996 1996. Available from http://www.broadcom.ie/~fs.

[46] J. E. Doran, S. Franklin, N. R. Jennings, and T. J. Norman. On cooperation in multi-agent systems. *The Knowledge Engineering Review*, 12(3), 1997. available at http://www.elec.qmw.ac.uk/dai/pubs/fomas.html.

[47] M. P. Wellman. A market oriented programming environment and its application to distributed multicommodity flow problems. *Journal of Artificial Intelligence Research*, 1:1–23, 1993.

[48] Pettie Maes. Agents that reduce work and information overload. *Communications of the ACM*, 37(7):31–40, 1994.

[49] Babak Esfandiari, Gilles Deflandre, and Joël Quinqueton. An interface agent for network supervision. In *?*, 1996.

[50] Malik Ghallab. Past and future chronicles for supervision and planning. In Jean Paul Haton, editor, *Proceedings of the 14th Int. Avignon Conference*, pages 23–34, Paris, June 1994. EC2 and AFIA.

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