# Network Management System with Intelligent Agents

# A first step with SLD

Pierre Conti, Jacques Labetoulle, Karina Marcus, Morsy Cheikrhouhou
Corporate Communication Department
Institut Eurecom
BP 193, 06904 Sophia-Antipolis Cedex
{conti,labetoul,marcus,cheikhro}@eurecom.fr

**Abstract**
Current Network Management Systems (NMS) are known as centralised systems. Some inherent known problems are lack of scalability, complexity to configure, network congestion, etc. In response to these issues, the Agent paradigm appears to be an interesting solution. The Distributed Intelligent Agents for Network Administration (DIANA) ([1]) project that we present, ***focus on the design of generic intelligent agents*** able to improve their behaviour by the acquisition of new skills

## I. Introduction

In a first section we **introduce the Intelligent Agent (IA)** paradigm, the different origins of agents explaining the confusion and debates about their definitions. In a second section we expose **why IAs are a solution** for current Network Management issues, the **advantage of the distributed management.** Then we propose the DIANA NM architecture, which is based on a hierarchical organisation, according to definition of management domains. In the following section we give a brief presentation of the DIANA agent architecture, the main components or layers that have been designed to fulfil the requirement of genericity and flexibility, and the related concept of skill modules. The last part is devoted to a case study implying DIANA agents managed from a web browser interface to explain the collaboration between agents. For this purpose, we use a scenario of fault detection, which explains the use of a System Level Diagnosis skill module. This example introduces also how several skill modules may be used to reach a same goal.

## II. Intelligent Agent, what is it ?

Intelligent Agents (IAs) seem to become a new fashion in computer domain as was the Objects a decade sooner. But IA concepts are less clear and a lot of different kind of agents has appeared recently. Software agents, personal agents, mobile agents, and Intelligent agents are often referenced by the press as sort of software in a great confusion. The reason of this mess, as we stated in [2], comes from the origins of what we call in this paper Intelligent Agents.
The three main origins we identified are firstly the Artificial Intelligence domain that has understood the interest of light software as opposed to the huge expert systems, secondly the Object Oriented domain that needs to give autonomy and mobility to its objects, and lastly the

[1] Project DIANA is financially supported by SwissCom

Network Management domain that owns agents since a long time, and needs to give them some intelligence. The term intelligence for a software is terrifying for many people who remember the problems that have been encountered in AI, but what is called intelligence here is a set of unintelligent properties that work together to provide an "relative intelligent behaviour". The common IA properties are:

- Autonomy: does not require instruction, knows what to do in what circumstances.
- Collaboration activity: shares work with other agents.
- Communication ability: knows how to exchange information with humans or other agents.
- Delegation: may ask on agent to do something for it.
- Deliberative: is able to reason according to its knowledge (beliefs).
- Learning capacities: acquires and uses new knowledge.
- Mobility: may move from a system to another and continues its activity.
- Planning: may organise its activity in function of priorities.
- Pro-activity: understand the cause of information and anticipate the effects.
- Reactivity: responds on event.
- Security: knows if an agent is corrupted or not.

If the co-ordination of unintelligent behaviours may serve to create an apparent intelligence, in the same way the co-ordination of agents as a whole system exhibits intelligent characteristics [4].
From the previous properties are designed agent architectures, which are classically:

- Deliberative:

based on a knowledge of its environment, and on reasoning capacity (logic programming).

- Reactive:

generally the agent has a model of its world with a set of predetermined actions to perform when it receives events.

- Hybrid (reactive/deliberative):

contains two layers one reactive one deliberative

All these difference explains what there is still no consensus about a definition of an Intelligent Agent.

## III.Intelligent Agents a solution?

Let see the three layers of a classical Network Management Systems architecture and its inherent problems.

### *I. Manager*

Its role is to collect and manage information reported by the management agent, to analyse them and to take decisions. Two known problems are:

**1) The amount of information to process** to extract some relevant information is very huge and therefore the NMS must use powerful database system that must be dimensioned to the network, causing problems if the network size increases. In some cases this increase implies that the NMS is to be changed for another or at least the hardware has to be changed.

Processing all these information requires applications, and these applications are more often complex to learn and to configure.
One of the main Artificial Intelligence goal is to develop systems able to "solve problem of the information complexity ... searching for relevant information in information space" [3]. They observed that it is more efficient to distribute the work to small intelligent applications that share they knowledge, instead of creating important expert systems. The idea to process the data directly (or close) to the source, is now taken in account by whose study the next generation of NMS, and IA are good candidates to do this work.

**2) The network bandwidth must be large** to support the concentration of all incoming events reported by the management agents. The congestion obliges a trade-off between the precision and the size of the management areas [5]. This problem disappears if the reported information is of a higher level of abstraction than the current raw data. Here also IAs have a role to play. Another advantage of using IA paradigm to develop NMS is the fault tolerance when a link goes down between the Manager and a part of the network. As the IAs are autonomous they may continue to manage the part of the network on which they are connected, and they avoid the deluge of alarm events that can devour network bandwidth.

### *II. Management Agent*

Management agents are in charge of performing operation on the managed object, and to report management information to the manager entity. But "the development of management agents is difficult, time consuming and ad hoc" [1]. This is why, NMS providers propose tools or framework to develop agent, based on generic architecture, or specific languages. But the new management agents developed are still not autonomous neither deliberative.

### *III. Managed Objects*

The managed objects describe the state of the network elements, and they depend on the protocol used mainly SNMP or CMIP. This obliges the development of protocol dependent agents. IA are protocol independents.

In this section we have seen some advantages to use IA paradigm, in the next section we present a view of what may be an Agent Oriented NMS.

## IV.DIANA NM Architecture

The architecture we propose is based on the extensible/flexible capabilities of IA. The agents are hierarchically organised with the possibility to share and delegate activities and responsibilities. Each agent has a NM domain, which represent a set of NE for specific NM activities. Their domains are different depending if, for example, they are doing Security or Fault management tasks. But an agent may owns and manage different domains at the same time. In the same way an agent X may be the manager of an agent Y for an NM area, and Y be the manager of X for an other NM area. This may be explained by the fact that agents are collaborative and the manager role is a responsibility role.

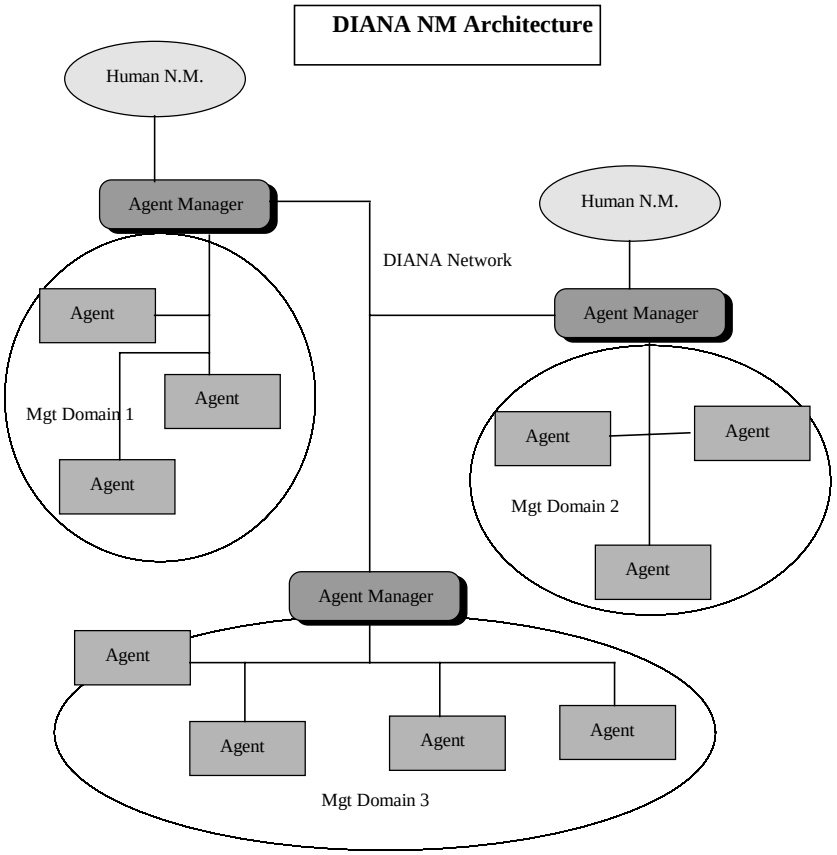


**Figure 1**

Agents communicate between them to exchange, information called beliefs , or objectives of management called goals, motivations, or policies.
To perform their tasks the agents need skills, which they may load from an other agent when they need them. The agents may be addressed by the users using a web browser if they have an interface skill, or with an other communication protocol as SMTP or even Telnet.

## V. DIANA Agent Architecture

We used an agent architecture of type hybrid (reactive and deliberative) keeping in mind the objective of being able to dynamically improve the behaviour of our IAs.
Several internals component are articulated around what is called the Brain. The Brain is in charge of analysing the interaction between the loaded skills, the incoming requests, planning the activity, and keeping "social" relationship with the other agents. The skill modules offer services to the agent's brain, and so to the other loaded skills. Different kinds of skill may be developed and used, as for example protocol skills which may access NEs having specific protocols, or high level skill management oriented, as for the supervising of Fault Management. When the Brain load a skill, it get the services the skill offers, the beliefs it uses or create, and the prerequisite other skill it needs to execute. So when a new belief has been created, the brain know and my forward it to all interested skill modules.

Intelligent Agent Concept

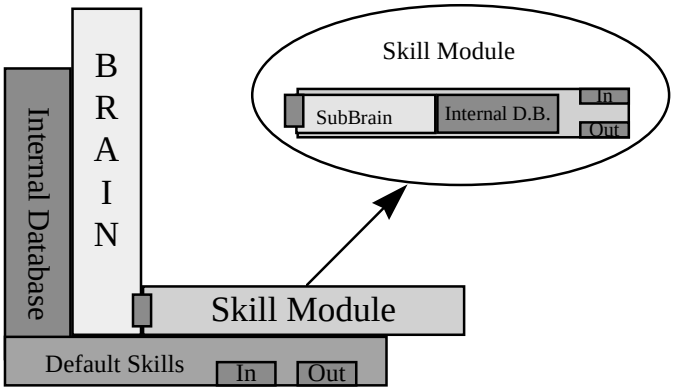


**Figure 2**

## VI.Use of Skill module, example with SLD

To validate our model of IA we have developed several skill modules that are expected to be useful for a NMS. The idea was to realise a management system able to continue its work even if one of its elements fails. The first developed skill module was an Agent Management skill, in charge of distributing domain of management to the available agent it knows. The second one was a monitoring skill that may access the network elements through the protocol SNMP. This is a simple polling mechanism which will perform on demand, and at the wish frequencies, SNMP get operation on NEs and will provide beliefs on the state of these NEs. The third developed skill module use a SLD (System Level Diagnosis) algorithm to verify the state of a distributed system. This algorithm uses the beliefs resulting of the monitoring activity performed by the IAs, and requires to these IAs to have some common representative elements monitored (fig 3). The next skill developed was a general Fault Management module, which just gets the domain attributed by the Agent Management skill, and asks every IA to monitor their NEs and report the results. The last skill is an Interface skill composed by the skill module itself and a java applet. Its objective is to start the Fault Management, display the monitoring results, force an agent to report erroneous information and force an NE (through the help of the monitoring) to stop any reporting.
The scenario used is the following:

- Start an agent on six different systems.
- Request one of the agent (will be the master agent) to load and start the interface skill module.
- Start a web browser on a system and call the agent to start the applet.
- Load and start the Fault Management activity to the previous designed master agent.
- Verify the state of all the NE

- From the applet force an agent to become unreliable
- From the applet choose and force a NE under the responsibility of the unreliable agent to go down
- Verify that the error is not reported (because the agent or the system it use has a failure)
- From the applet ask the master to load and start the SLD skill module.
- Verify that the agent is detected as in state unreliable and that its domain is redistributed.

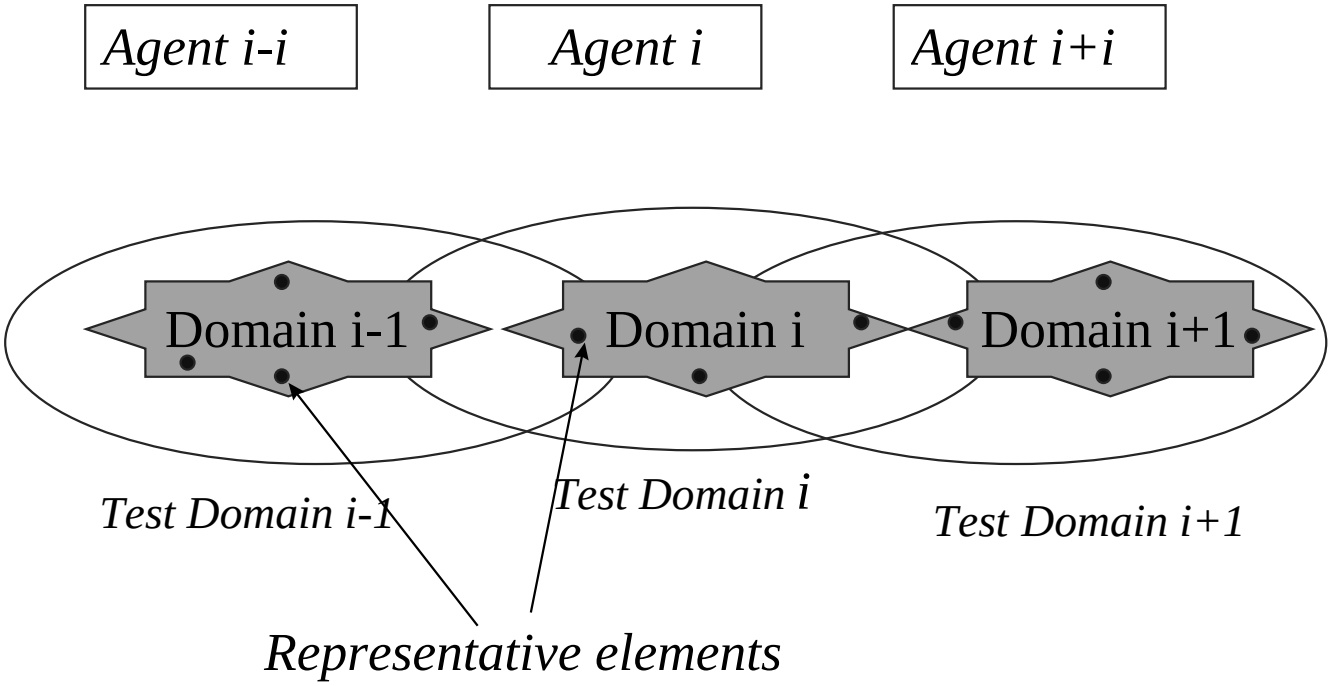


**Figure 3**

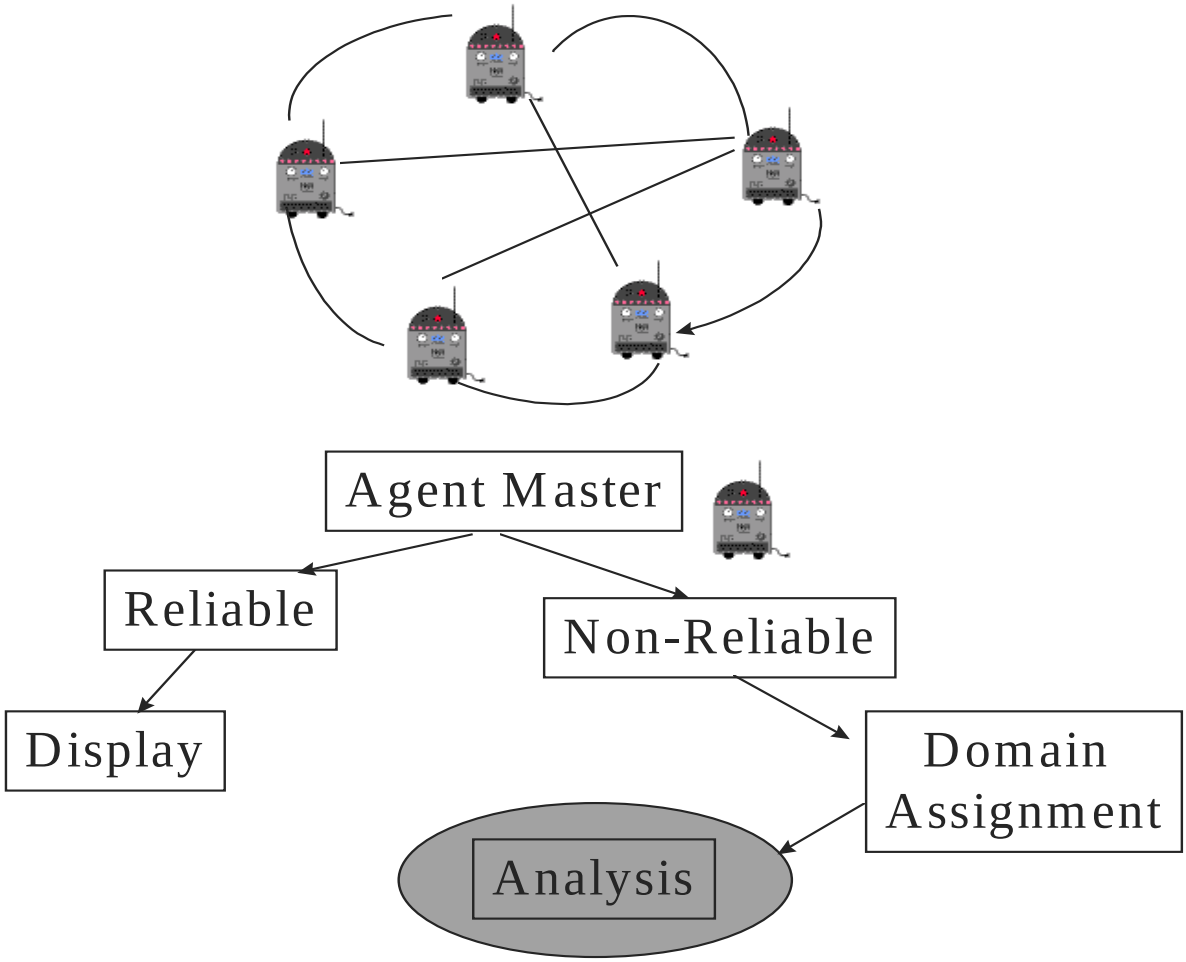


**Figure 4**

In this test, the Fault Management didn't perceived that an NE was down until the SLD was started. At this moment the SLD diagnoses that an agent is unreliable and create a belief on the master agent expressing that an agent is unreliable. This information is forwarded by the master's Brain to the Agent Management skill which understand that it has to redistribute the agent domain (NEs) of this agent to the other available agents. The Fault Management skill has been also advertised of wrong state of the agent and of the domain redistribution. Also it requires the agents to start the monitoring of their new NEs. At the end, the monitoring may continue correctly. As soon as the unreliable agent will be back in a good shape, it will take back its monitoring activity.

## VII.Conclusion and further study

In this implementation we have demonstrated that a DIANA agent system may be easily used to monitor a network and perform failure recovering. As the agent are distributed, close to the NEs they have to manage and able to report only useful information, saving bandwidth consumption.

The following step will consist to improve the belief language taking in account the NM specificity, and to develop a full set of NM skills to cover all Network Management areas as

for example a topology skill module that will be used to analyse and localise the faults and for configuration.

**References**

[1] G. S. Perrow, J. W. Hong, M. A. Bauer and H. Lutfiyya,. "The Abstraction and Modelling of Management Agents". In Proc. of the Fourth International Symposium on Integrated Network Management, Santa Barbara, CA, May 1995.

[2] M. Cheikhrouhou, P. Conti, J. Labetoulle ."Intelligent Agents a State-of-the-art". Submitted to Networking and Information Systems Journal. 1998

[3] J. A. Hendler. "Intelligent Agents: Where AI Meets Information Technology". In IEEE EXPERT Vol. 11, No 6: December 1996.

[4] Jorg P. Muller. The Design of Intelligent Agents. A layered Approach. In Lectures notes in Artificial Intelligence 1177. Springer

[5] Fergal Somers. Hybrid: Unifying centralised and distributed management for large high-speed networks. In Networks Operation and Maintenance Symposium (NOMS96), Kyoto, 1996.