Intelligent Agents for Network Management: Fault Detection Experiment *

M. Cheikhrouhou, P. Conti, J. Labetoulle, K. Marcus
Corporate Communication Department
Institut Eurécom
BP 193 – 06904 Sophia-Antipolis Cédex – France
{*cheikhro,conti,labetoul,marcus*}*@eurecom.fr*

Abstract

We present an architecture for flexible intelligent agents dedicated to build network management systems and its application for a concrete experiment. The agent architecture is based on Skills that can be plugged on an agent Brain. The skill is a piece of code that can be loaded seamlessly into an agent to supply it with more new capabilities. The brain is a common part that governs the whole function of the agent.
We detail how this architecture was used to build a fault-tolerant agent system that distributedly monitors a multi-domain network. The fault tolerance was insured by a detection mechanism based on System Level Diagnosis, and a recovering procedure based on task redistribution. The experiment showed how the agent system could ensure the distributed monitoring even in the case of the partial failure of the agent system.

Keywords

Distributed Network Management, Intelligent Agents, Fault Management.

1. Introduction

Network Management (NM) is becoming increasingly complex due to the embedded increasing complexity of the network elements multiplied by the increasing size of the

*This research was supported by Swisscom and by the Institut Eurécom.

network, the needs of more sophisticated services, and the heterogeneity challenges.

Therefore, new Network Management Systems (NMS) must be:

- flexible to deal with the fast and frequent configuration changes;
- scalable to improve the profitability of the huge investment that represents its installation, configuration and launching;
- simple to decrease the learning delay and reduce the human expertise needed to use it;
- reliable, and therefore able to anticipate the risk of failure, and to continue to work even if part of the network is cut out of the NMS;
- and secure.

There are several approaches that have been studied to tackle these requirements. Among these, we believe the Intelligent Agent (IA) approach is the most interesting and promising. This idea seems to be shared by the Network Management community if we observe the progressive number of papers about IA during NMS congresses [1, 2].

There are two main trends that make use of the term Intelligent Agent. The first one works on improving legacy network management agents, like SNMP agents, by more processing capabilities, and then call them IAs [13, 14]. The second trend, in contrast, is applying Intelligent Agent techniques borrowed from the Distributed Artificial Intelligence (DAI) and Multi-Agent Systems (MAS) domains [12].

Within the DIANA project, we are more committed towards the second trend. The main objective of the DIANA project is to give solid bases of a multi-agent system, which shall be easy and simple to use and program, offering light service primitives that can be used in a fast and efficient manner to form powerful re-usable blocks. For us, the term agent refers to that used in the DAI domain and not to that used in the NM paradigms such as CMIP or SNMP agents.

In Section 2. we shortly review the concept of an intelligent agent, its main properties and architectures. Next, we motivate the distributed network management using intelligent agents in Section 3.. The DIANA architecture is presented in Section 4., and in Section 5. we describe how this architecture is used to develop a robust and reliable monitoring system. We close the paper with some conclusions and directions for future research.

2. Intelligent Agents

Intelligent Agents seem to become a new fashion in computer domain as was the object paradigm a decade ago. However, there is little agreement about the definition of what an agent could be and what architecture is most suitable to use [10, 15]. Of course,

such debate is out of the scope of this paper. Instead, we present in the following our view of the concept of "agency" as adopted in our work.

Viewed as a blackbox, an agent is a persistent software that acts autonomously on behalf of its user. An agent is expected to have the following properties: autonomy, communication, cooperation, responsiveness, pro-activeness and learning.

Autonomy means that the agent has control over its actions and over its own state, decides by itself what action to do under which circumstances and is able to accept abstract high-level missions and to ensure their achievement. Communication means that the agent can communicate with the other agents and with its user. Collaboration means that a set of agents carry out tasks cooperatively, possibly using delegation or coordination mechanisms. Responsiveness requires that the agent reacts in a timely fashion to possible changes in its environment in a way to ensure the achievement of its duties. Pro-activeness is even a stronger property and means that the agent is able to anticipate changes in the world to perform preventive actions or to take the opportunity to achieve more quickly or more efficiently some goal. Finally learning means basically the ability to improve its behavior in time and to induce new knowledge of its environment during its operation.

Mainly, there are three kinds of agent architectures [10]:

- Deliberative: based on a knowledge and a symbolic model of its environment, and on reasoning capacity (logical programming and planning).
- Reactive: generally, the agent has a set of pre-determined actions to perform when events occur in a stimulus-reaction way.
- Hybrid: combines both reactive and deliberative architectures.

The interest that more and more research teams are showing towards investigating the IA paradigm to deal with the complexity of network administration applications is motivated by the possibility to distribute the information over the network, the small number of communication exchanges, the capability of acting/reacting to events, and the possibility to reason on the local data, implying a possible pre-processing of information before a global consolidation takes place.

We insist on the fact that the concept of *agent* is used both in the DAI and in the NM domains with strictly different meanings. In the following, we use the intelligent agent concept as defined by the DAI community, unless explicitly specified, e.g. SNMP Agent.

3. Distributed Network Management Using Intelligent Agents

The distribution of NM activities has many advantages, such as to decrease network congestion around the NM console due to the management traffic, or to reduce the amount of information to process and to augment the fault tolerance of the NMS [16].

Several solutions have been investigated, some being to enhance the SNMP agents and some being to use a hierarchical NMS architecture or both.

The use of a proxy (or intermediate) agent with some reasoning capability has been proposed in [5, 7, 13, 14]. Usually in this approach, a MIB is designed to contain control commands [5, 7] or rules [14]. In [5, 7, 13], the agents execute scripts sent by the manager. SNMP is used in [5, 7] to upload the scripts on the agents, and to bring the results back to the manager.

Notice that in the above mentioned works, Intelligent Agent properties like cooperation and deliberation are not used. Concerning studies steaming from the Intelligent Agent theory, in [4] one can find a comprehensive presentation of the state of the art.

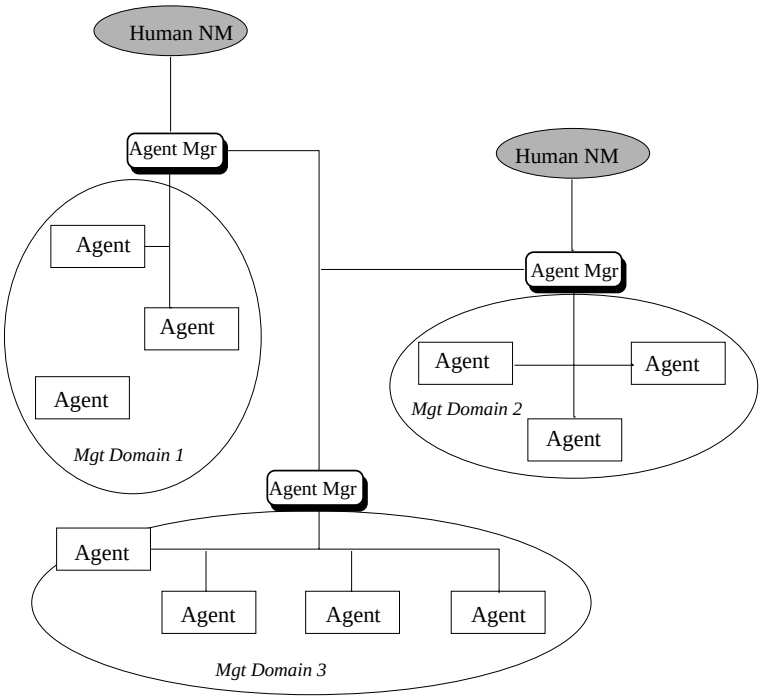

Figure 1: Agent Distribution in the DIANA Architecture

The architecture we propose in the DIANA project is based on the extensible/flexible capabilities of intelligent agents, on top of their known properties presented in section 2.. The agents are hierarchically organized with the possibility to share and delegate activities and responsibilities. Each agent has a NM domain that represents a set of network elements (NEs) for specific NM activities (see Fig. 1). Their domains may be different depending on their tasks (security, fault management, etc).

The main principle in the DIANA NM architecture is that network management functions can be identified and implemented as *skills*. For example, a skill may have

a fault detection function, an accounting or monitoring function, etc. Obviously, skills may use other skills. Agents may then be more or less skilled in one management domain. In order to delegate tasks or to work in a cooperative way, the agents need to load or may be asked to load some special skill, which may then be discarded once the task is accomplished. In the next section we discuss more in details how the skill enters in the composition of a DIANA agent.

In Section 5. we show how the hierarchical organization may be implemented with DIANA agents, and how cooperation and delegation appear as important and useful properties to network management using agents. Examples of skills will also be presented to illustrate the properties of extensibility and flexibility.

4. DIANA Agent Architecture

In our agent architecture, focus has been conducted towards flexibility and dynamicity. Agents are wanted to be able to aquire new capabilities and skills seamlessly without interrupting their operation. This is essential for network management purposes where network elements may need to be upgraded frequently and therefore, the management application needs to be easily adaptable. For these reasons, our agent architecture is based on two major component types: the Brain and the skills. Skills provide the agent with capabilities and behaviors, while the Brain is the "headmaster" that accepts and manages agent skills.

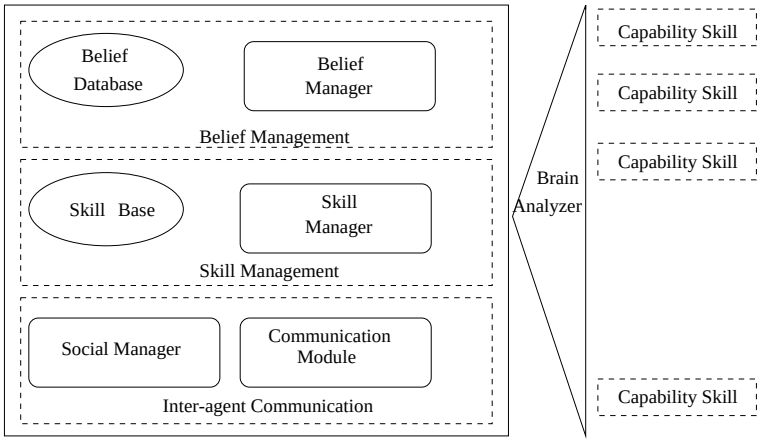

Figure 2: DIANA agent architecture

4.1 The Agent's Brain

The Brain (Figure 2) offers basic and innate facilities necessary for the agent operation. These facilities are either local facilities, i.e. for the agent's local operation or inter-

agent facilities, i.e. responsible for agent communication and interaction with the other agents.

Local Facilities

Locally, the Brain is reponsible for maintaining the agent's belief database. The concept of *belief* in the agent community is known as (see [9]) "the agent's expectations about the current state of the world and about the likelihood of a course of action achieving certain effects". In the context of network managment, agent beliefs hold the management information the agent has, as well as information about the other agents it knows about. For example, the status of a switch that the agent is managing can be expressed as a belief: `(switch :name sw301 :ipAddr 193.55.63.48 :status Ok)`. The agent's beliefs can be accessed concurrently by several skills during the agent operation. Therefore, the Brain includes a *Belief Manager* that ensures the coherent access to the belief database and maintains its integrity.

The Brain is also reponsible for the management of the agent's skills. Skills can be downloaded on-the-fly and integrated into the agent inside its *Skill Base*. If a loaded skill is no more useful for the agent operation, it can be disposed off so that to keep the agent's size as small as possible. Newly loaded skills may require pre-requisite skills, and the Brain is reponsible for checking whether these skills are available and are already loaded into the agent or not. If a skill is necessary for another one, and is not yet loaded, the Brain is responsible for looking for it either locally or via the help of some other agents.

When a skill becomes active, it makes use of the agent's beliefs by creating new ones, updating or deleting existing ones. A skill operation may depend on beliefs created by other skills, and the Brain is therefore in charge of dispatching asynchronously these beliefs to the interested skill in a transparent way. These facilities are provided by the *Skill Manager* which holds the necessary information about the skills in the *Skill Base*.

All these three functions are governed by the *Brain Analyzer* whose role is the parsing of the messages that the brain receives, either from the skills or from the inter-agent communication.

Inter-agent Facilities

The Brain offers also inter-agent communication facilities that allow skills from different agents to interact in a transparent way. A *Communication Module* inside the agent is responsible for sending to and receiving requests from the other agents. Another module, the *Social Manager*, holds information about the other agents, such as the host on which they run as well as its address and access port. Therefore, skills only deal with the symbolic names of the distant agents they want to interact with, and they are not aware of network-level details of the agent system configuration.

4.2 Skills

An agent skill is a piece of software specialized in a network management area and that can be plugged-in dynamically into the agent. The skill enriches the agent with a new set of capabilities. A capability is a new type of action and processing the agent becomes able to execute after loading a skill. For example, an SNMP skill would allow the agent to initiate SNMP primitives such as `get` and `set` operations. Usually, a skill offers new services and more elaborated beliefs to other skills. For this, it may use the beliefs and the capabilities offered by other skills that are supposed to operate at a lower level.

The skill has an interface that communicates initialization information to the Brain. This information declares the pre-requisite skills it needs for its operation and the set of capabilities it offers. Each capability declares the beliefs it requires, the capabilities that may be required from other skills and the created beliefs. During skill initialization, these information allow the Brain to determine whether all the necessary pre-requisite skills are available or not. During the skill operation, the Brain can determine which skill is concerned by a capability invocation in order to forward it to the corresponding skill. Furthermore, it is able to determine, according to which capabilities are currently requested from that skill, which beliefs should be dispatched to the skill as they are created, updated or deleted.

For example, a domain management skill may be used within an agent system to ensure the domain assignment of management tasks among the agents. This skill may offer a capability to load-balance the management tasks between the agents. This capability requires beliefs on the agents telling which agent is better suited to accept a new management task. It also produces beliefs telling which management task is affected to which agent.

The interaction between the Brain and the Skill may either concern beliefs or capabilities. A Skill can request the Brain to create, delete and update beliefs and can query for a particular type of beliefs. The Brain notifies each Skill of the pertinent changes that the other skills make on the belief database. For example, the Brain notifies the domain management skill of each change on the beliefs concerning the operational statuses of the other agents. If an agent crashes, then the domain management skill is notified and can take the corrective actions, i.e. to reassign the tasks that have been affected to the crashing agent to the other agents in the system.

Concerning capability interaction between the Brain and the Skills, a Skill can ask the brain to execute an action corresponding to a certain capability. The brain is therefore responsible for finding the Skill that contains this capability. When such a Skill is found, the Brain invokes the requested capability after checking whether the required capabilities and beliefs are available or not.

5. The Experiment

5.1 Experiment Purpose

We consider a corporate network composed of a number of sub-networks. In order to manage the whole network, it is straightforward to assign an intelligent agent for each subnetwork. This strategy allows to have a fine grain management of every sub-network where the bandwidth is not a concern.

Usually, such approach is not fault-tolerant. In effect, when an agent crashes, the sub-network it is affected to is no more manageable. This can be annoying when it is no more possible to manage sensitive elements in this sub-network. Moreover, since the agents are supposed to perform autonomously, it may take a long time before discovering that an agent has crashed.

Therefore, one approach to have a fault-tolerant management system based on autonomous intelligent agents is to include a mechanism that allows to detect crashed agents. This mechanism should allow the other agents to be informed promptly of the crash. When the other agents are informed, they have to take the initiative to undertake the responsibility to ensure the tasks that were performed by the crashed agent. The latter may be repaired later, for example after a human intervention. The agent system should be able to detect its recovering and to reassign the management tasks in a more optimized way.

In our experiment we suggest to build a prototype of such a fault-tolerant management system based on the DIANA agent architecture. As a fault detection mechanism, we propose to use the System Level Diagnosis, which will be presented later in this section.

As an example of distributed management task the agent system may achieve, we choosed the monitoring of network elements for fault detection purposes.

5.2 Scenario

We propose a simplified scenario in which the human network administrator is asking for a high-level goal of monitoring the sensitive network elements, e.g hubs and switches, to detect the faults that may occur. He connects via an applet to a particular agent that collects the required data. These data are reported by intelligent agents affected to the domains of the managed network. Each agent performs fine-grain and detailed monitoring of the sensitive network elements of its domain in order to come up with the high level global status of each element, which is then reported to the agent to which the administrator is connected. For commodity, we call this agent the *Master Agent*, while we call the other agents *Domain Agents*.

The applet used for the experiment demonstration is shown in Fig. 3. At the left side, it shows the statuses of the monitored network elements, each element being associated to the domain agent to which it is affected. The right side of the applet is used both to control the demonstration via the control buttons (in the upper panel), and to dump the beliefs that were received by the Master Agent (in the lower panel).

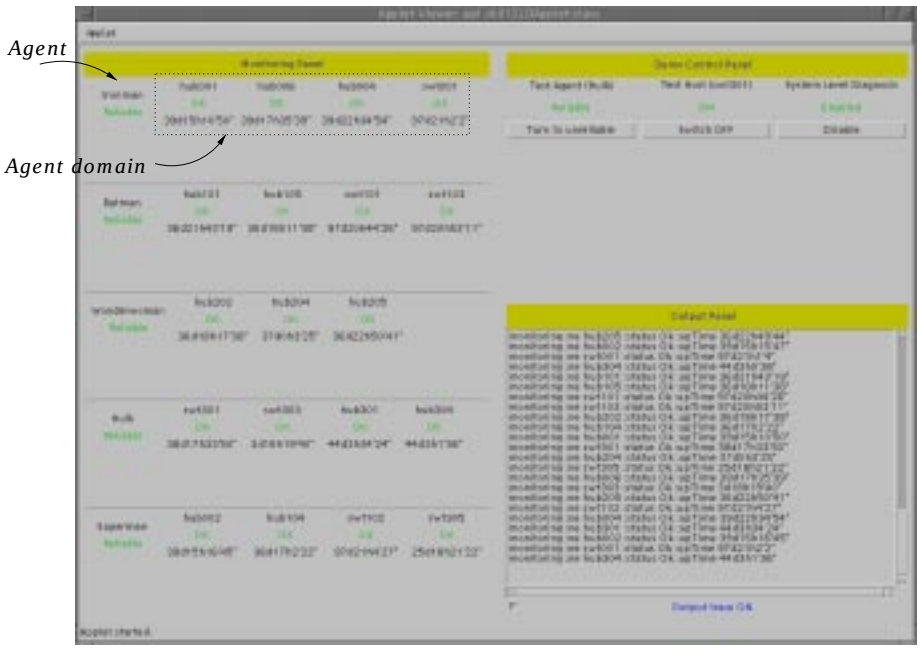

Figure 3: Interface panel

When one of the five domain agents crashes, the master agent detects it thanks to the SLD algorithm. It then reaffects the elements that have been monitored by this agent to the other agents in the system. Hence, the monitoring continues as if there were no crashed agent, and the network administrator is still delivered with the monitoring information he asked for. Later, the crashed agent may recover. Again, thanks to the SLD algorithm, the master agent detects it and affects the monitoring tasks that had been previously affected to it.

5.3 Required Skills

In our experiment we wanted to implement an agent-based distributed NMS able to perform monitoring and fault detection activities – *even* if one of the agents is faulty. To ensure reliability of the information sent by each agent – meaning that the agent is fault-free – a system level diagnosis (SLD)[8] algorithm is used.

A Master agent is designed to centralize the monitoring activity, and other agents – called *domain agents* – wait for the Master commands. The Master is in charge of assigning domains to domain agents, to delegate the task of monitoring and to verify the reliability of each agent before displaying the monitoring information. If one agent is found to be faulty, then the Master will reassign the domains, so that the monitoring may reliably continue; if one unreliable agent recovers from its failure, then the Master

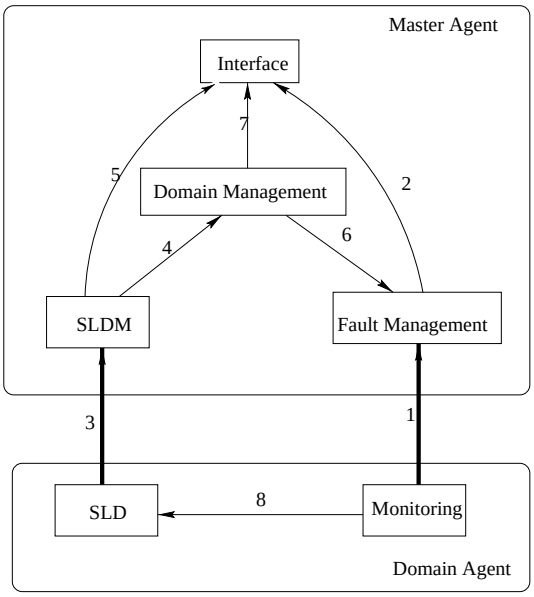

Figure 4: Skill Interaction Diagram

reassigns the domains to balance the global workload of the domain agents.

All these tasks are executed by the agents through skills, which will be discussed later in the coming sub-sections. In the description of each skill the reader will find indications about the information and belief flow between the skills. These indications, e.g. ①, refer to the arrow numbers in the diagram presented in Fig. 4. In this diagram we depict the hierarchical composition of the NMS, and detail in each agent the skills that are involved in the information exchange. Notice that the bold arrows represent inter-agent communication, while the thin arrows – representing skill interaction – are simple brain notifications about belief changes.

The proposed agent implementation relies on six different skills: four that run in the Master agent and two in the domain agents. The Master agent runs an Interface skill that supports the GUI provided through applets. The Agent Management skill is responsible for the domain affectation among the domain agents, and the Fault Management skill uses this affectation to delegate monitor tasks accordingly. The SLDM skill is the centralized part of the SLD algorithm. It uses the SLD skill that runs upon each domain agent and that uses by its turn the Monitoring skill. The following sub-sections detail these six skills.

Domain Management Skill

This skill is loaded into a particular agent, called the Master Agent, to make it responsible for the domain managment in the agent system. The domain management skill is notified whenever a change in the beliefs about the reliability of the domain agents occurs ④. If this is the case, the Domain Management performs a domain re-distribution in a way that ensures that all the requested management tasks are still performed and are well distributed amongst the operational agents. This causes the beliefs about the domain affectation to change, which will in turn cause the brain to notify the fault management skill ⑥ and the interface skill ⑦.

The belief corresponding to the domain affectation has the following pattern:

```
(AgentDomains :agent Ironman :domain (hub101 hub102 hub201)
              :agent Hulk : domain ()
              :agent Batman :domain (hub301 hub303 sw202)  )
```

Let's insist on the fact that being a Master does not require a dedicated agent. Indeed, thanks to the possible dynamic loading of skills, any agent in the agent system can potentially ensure the role of the master by loading the necessary skills, i.e. the domain management skill, the fault management skill, the interface skill and the SLD master skill. In fact, it could be possible to make a domain agent ensure at the same time the functions of the master agent.

Fault Management Skill

The Fault Management skill is loaded in the Master agent. It is responsible for delegating the monitoring tasks to the domain agents. The delegation process is based on the domain affectation ensured by the Domain Management skill ⑥. The domain agents send the monitoring information ① to the Master agent and this information is then made available to the Interface skill to be displayed ②.

Monitoring Skill

The monitoring skill is responsible for instrumenting the necessary beliefs about the monitored hosts. This information is then used by the SLD skill ⑧ and the Fault Management skill ①.

The general scope of the monitoring skill is the monitoring of network element (NE) information in a protocol-independent way. It looks for the best way to instrument these parameters and decides of the way getting these information is best performed (polling and polling frequency, traps, SNMP requests or RPC commands, etc.). The monitoring skill is able to produce a summary of the global status of a network element from a detailed monitoring procedure.

System Level Diagnosis Skills

This skill module has been designed to allow a system of agents to ensure a reliable behavior from each agent in the system, using the well known method called system level diagnosis (SLD). [3, 6]

The SLD operation is centrally controlled by the Master agent. A master skill (SLDM) will be responsible for deciding which agents are reliable [8]. Every other domain agent executes a slave SLD skill, which is in charge of the testing.

Based on a test graph, the SLDM skill assigns to each domain agent a set of neighbor domain agents. To execute the test, the SLD skill accesses beliefs produced by the monitoring skill ⑧ about some chosen network elements – the representative elements – in the agent domain and in the domain of this agent neighbors. The domain agents exchange the collected information and compare the local and distant ones in order to diagnose their neighbors. Afterwards, each SLD skill sends the test conclusions to the SLDM skill ③, which runs a diagnosis algorithm on the conclusions and creates beliefs reflecting each agent's status. The Interface skill ⑤ and the Domain Management skill ④ are notified of this information.

The SLD theory stipulates that a correct diagnosis about the agents can be established as soon as no more than half of the agents are faulty [11] (and we consider that this assumption is always satisfied).

Interface skill

The interface is in charge of linking the agent's world to the human world. The user, typically the network manager, uses his WEB browser to access the agent, giving goals and receiving results or notifications.

In this experiment the Interface skill provides (see Fig. 3):

- multiple access to the Master agent from any standard Java-enabled browser,
- the demand of agent predefined management goals,
- the display of the Master agent information on the execution of the goals,
- network monitoring information per domain agent,
- the status of each agent.

5.4 Experiment Results

Eurecom's network was used as a testbed for the experiment. The agent system was used to monitor sensitive elements of the network such as hubs and switches. Eurecom's network was composed of five sub-networks, each of which was affected to one of the domain agents, as indicated in Fig. 3. In addition, we used another agent as a Master to collect the monitoring information and to manage the agent domains.

The master agent also allowed the network administrator to connect via an applet that displays the monitoring results. Many applets could connect at the same time.

The experiment was composed of two stages: with and without the SLD mechanism. At the first stage, the monitoring was launched on the five domain agents which start to forward the results to the master agent. We cause the crash of a domain agent (this agent was *Hulk* in the demonstration). The crash was not detected by the agent system. Also, the monitoring of the network in Hulk's domain was no more ensured. The results displayed on the applets about Hulk's domain were out-of-date.

At the second stage, the SLD mechanism was enabled. As previously, the monitoring was launched on the domain agents and the applet starts to display the monitoring results. Afterwards, we caused the domain agent *Hulk* to crash. In this case, the crash was detected as soon as the master agent received the SLD beliefs from the domain agents and ran the SLD algorithm. The SLD Master skill creates the belief: `(agent :name Hulk :status unreliable)`. The brain notifies the domain management skill which caused the master agent to perform a domain re-distribution. The network elements that had been previously monitored by *Hulk* were affected to the other domain agents in a balanced way. Therefore, the network administrator could receive the monitoring information despite of the failure of a part of the agent system. This property introduced an aspect of fault tolerance into the network management system.

6. Conclusion and Further Work

The fact that Intelligent Agents are more and more present in the Network Management area is undeniable. The problems of the actual NM paradigms let one think that the properties of intelligent agents together with extensibility are a good recipe to deal with a more hierarchical and decentralized management. As far as it goes, the DIANA architecture seems to be in the right direction.

In this work, we introduced the main ideas of the DIANA Intelligent Agents, showing how an intelligent agent system may be used in the NM area. A hierarchical and distributed model of NM is given using intelligent agents, and the main components of the basic agent are discussed. To show the feasibility of the DIANA technology, a reliable monitoring experiment was presented, where agents specialize their capabilities by aquiring different skills.

The experiment we presented revealed the great advantage of using skills to implement NM functions. Due to the information model based on belief exchange we could easily develop the skills. The flexible format of the beliefs allows the designers to communicate only the needed information, in a quite free style. Of course, a good specification is necessary to design a skill, but all the services provided by the intelligent agent's Brain are powerful tools that greatly simplify the design process. In addition, the facility of inter-agent communication allows an easy implementation of delegation schemes, bringing hierarchical and decentralized NM forth.

The experiment showed also how to provide a fault-tolerant management system

using distributed intelligent agents. To detect failures in the agent system, we used a variation based on the System Level Diagnosis. SLD has the advantage of testing distributed complex systems in an abstract and modular way. The recovering procedure is a simple reassigment of management tasks among the reliable agents.

To enhance the features of the DIANA Intelligent Agents, it is planned to improve the interaction between the Brain and skills by implementing: belief registration (to inform other agents that it will be interested on this or that type of belief), belief filters, temporal properties of beliefs, belief operators, a programmable reactive layer, etc.

In order to measure the capabilities and usefulness of the DIANA technology, we intend to pursue some other case studies, including:

- Security. A security module is being implemented using the DIANA architecture. The goal is to monitor unsuccessful login trials in order to detect possible attacks. The cooperative property of agents is used in this case to communicate partial results and to distribute the information.
- Application to the ATM technology. We are using Intelligent Agents to control and manage Permanent Virtual Circuits establishment in order to emulate Switched Virtual Circuits that support Quality of Service contracts.
- Topology. A topology module will be implemented to enable analysis and localization of faults, and to help configuration.

Most probably other works will also show that intelligent agents may be a very good solution to the current NM challenges.

References

[1] *5th IFIP/IEEE International Symposium on Integrated Network Management*, volume V. Chapman & Hall, 1997.

[2] *IEEE/IFIP 1998 Network Operations and Management Symposium*, feb 1998.

[3] G. Berthet. *Extension and Application of System-level Diagnosis Theory for Distributed Fault Management in Communication Networks*. PhD thesis, École Polytechnique Fédérale de Lausanne, Lausanne, CH, 1996.

[4] M. Cheikhrouhou, P. Conti, R. Oliveira, and J. Labetoulle. Intelligent agents in network management: A state-of-the-art. *Networking and Information Systems Journal*, 1:9–38, jun 1998.

[5] G. Grimes and B. P. Addley. Intelligent agents for network fault diagnosis and testing. In *5th IFIP/IEEE International Symposium on Integrated Network Management*, pages 232–244, 1997.

[6] S. Hakimi and K. Nakajima. On adaptive system diagnosis. *IEEE Transactions on Computers*, c-33(3):234–240, Mar. 1984.

[7] P. Kalyanasundaram, A. S. Sethi, C. M. Sherwin, and D. Zhu. A spreadsheet-based scripting environment for snmp. In *5th IFIP/IEEE International Symposium on Integrated Network Management*, pages 752–765, 1997.

[8] K. Marcus. Improving reliability of intelligent agents for network management. Pre-print, 1998.

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

[10] H. S. Nwana. Software agents: An overview. *Knowledge Engineering Review*, 11(2):205–244, Oct. 1996.

[11] F. P. Preparata, G. Metze, and R. T. Chien. On the connection assignment problem of diagnosable systems. *IEEE Transactions on Electronic Computers*, EC-16(6):848–854, Dec. 1967.

[12] A. Sahai, C. Morin, and S. Billiart. Intelligent agents for a mobile network manager. In D. Gaiti, editor, *Intelligent Networks and Intelligence in Networks*, pages 449–463. IFIP, Chapman & Hall, 1997.

[13] M. Suzuki, Y. Kiriha, and S. Nakai. Delegation agents: design and implementation. In *5th IFIP/IEEE International Symposium on Integrated Network Management*, pages 742–751, 1997.

[14] M. Trommer and R. Konopka. Distributed network management with dynamic rule-based managing agents. In *5th International Symposium on Integrated Network Management*, San Diego, may 1997.

[15] M. Wooldridge, J. P. Müller, and M. Tambe. Agent theory, architectures, and languages: A bibliography. In M. Wooldridge, J. P. Müller, and M. Tambe, editors, *Intelligent Agents II*, volume 1037 of *Lecture Notes in Artificial Intelligence*, pages 408–431. Springer Verlag, 1996.

[16] Y. Yemini, G. Goldszmidt, and S. Yemini. Network management by delegation. In *2nd International Symposium on Integrated Network Management*, pages 95–107, Washington, DC, apr 1991.