Institut EURECOM
2220, route des Crêtes
B.P. 193
06904 Sophia Antipolis
FRANCE

Research Report N.o 98-045

Improving Reliability of Intelligent Agents for Network Management[1]

K. Marcus

December 1998

Name:	Telephone:	E-mail
K. Marcus:	+33 93 00 26 55	marcus@eurecom.fr
Fax:	+33 93 00 26 27	

[1]Eurecom's research is partially supported by its industrial partners: Ascom, Cegetel, France Telecom, Hitachi, IBM France, Motorola, Swisscom, Texas Instruments, and Thomson CSF.

Abstract

The Intelligent Agent technology is being more and more used in treating all kind of information; in the case of network management, where machines monitor machines, a special care must be taken with respect to the reliability of the information. In this paper we discuss how to ensure reliability using the System Level Diagnosis. A diagnosis test specially designed for the network management using a multi-agent system is presented, and we show how the SLD technique was implemented in the scope of the DIANA project.

Keywords: distributed network management, intelligent agent, information reliability, system level diagnosis

Contents

1 Introduction

Intelligent Agents are becoming a new fashion in computer domain as it happened with the Object Paradigm several years ago. Different kinds of agents have appeared with different labels and functions, and many authors have already tried to classify the growing population of agents [15, 19, 24]. One of the objective of this new trend is to give an apparent intelligence to a light software, compared to the huge expert systems.

In the DIANA[2] (Distributed Intelligent Agents for Network Administration) project, the main objective is to achieve network management using Intelligent Agents delegating tasks in a simple and easy way, using mainly the deliberative and reactive properties of agents [9]. Moreover, the DIANA agents must be able to communicate and "cooperate" to achieve in a coordinate way the goals they are given.

Delegation is not a new idea in management systems (see [25]). Other researchers and projects have already been interested in using Intelligent Agents for network management. For instance the authors in [22] suggest mobile agents to decentralise network management, in an extension to the client/server model in which the client and server exchange messages during execution. In [23], a multi-agent system (HYBRID) is proposed for traffic management in ATM networks

The network manager that uses an Intelligent Agents system must trust his agents. He has to be sure all the time that each agent is running and that it has correct and updated information. Of course in most of the distributed agent systems this should be true, but within the network administration domain this is a condition *sine qua non*, since the information collected is used to control the network, to diagnose the connectivity and security states, to change configurations and to improve the quality of service offered to the users or to achieve any goal delegated to them.

Faults in an Intelligent Agent system may occur: hosts going down, congested links, CPU overload, etc. and even the agent code may be corrupted. If an agent is not capable of accomplishing its goals, then either it must be replaced, repaired or the goals he has to achieve should be redistributed to the other agents. Hence, in order not to ruin the management system, the network manager and the system as a whole must be aware of the condition of its components, every possible faulty agent (where the fault may be caused by the machine, the environment or any other reason) must be discovered as soon as possible, and the actual set of "faulty-free" agents must be known.

Some studies in failure detection and recovery in multi-agent systems have already begun [1, 16]. In the approach adopted in [16], the goal is to verify if the behavior of one agent is coherent with the global goal, using social comparisons. The authors in [1] propose a data distribution to facilitate recovery, based in a temporal communication scheme. In the present work, the objective is to consider not only the properties of the

[2]Project DIANA is financially supported by Swisscom.

multi-agent system, but also to take in account that this system is dedicated to network management tasks. Therefore, "fault-free" is understood here as reliable.

To find out which agents are reliable or not, one must diagnose the Intelligent Agents Network. Many different diagnosis models exist, for instance: model based diagnosis [11], alarm correlation [5, 17], probabilistic methods [12], expert systems and case based reasoning [6, 10, 18] and the System Level Diagnosis (SLD) model. Due to the specific characteristics of the Intelligent Agent system case, the SLD was chosen to update continuously the knowledge of the reliability degree of the Intelligent Agent system.

The contents of this article are the following. The choice of SLD will be discussed in Section 2. In Section 3 the concept of reliability will be defined. The testing scheme will be presented in Section 4. To be able to use SLD within the intelligent agent model, we indicate in Section 5 how the diagnosis was implemented within the DIANA agent architecture. In Section 6 we discuss how all the information required for the diagnosis can be used for a further reasoning about the networks state. Finally we close the paper with some remarks and research directions. In what follows we use *agent* and *intelligent agent* interchangebly.

2 Diagnosing model

The System-Level Diagnosis model was introduced in [20] to diagnose faulty units in a system. In a nutshell SLD considers each node in a network (in the present case, each agent) as a unit. One unit may test and be tested by the other units, each test is assumed to produce a binary result: faulty or fault-free. The result of all the tests is called the **syndrome**, and the diagnosis is done using the obtained syndrome.

Much research has already been done concerning SLD, and among the possible approaches, we can find:

- centralized [3, 7, 14, 20] and distributed algorithms [2, 4, 8, 13, 21]
- one-step [3, 8, 20] and adaptive diagnosis [2, 4, 7, 13, 14, 21]
- faulty nodes diagnosis only [2, 3, 4, 7, 14, 20, 21], faulty nodes *and* links diagnosis [3, 7, 13].

As it was mentioned in the introduction, the goal is to provide knowledge about the reliability of the system, that is, reliability of each agent must be tested. We will consider the Intelligent Agent system as the virtual network composed of agents and links between agents, where the virtual links are implemented through physical wires, cables, paths using routers, bridges, etc. In this case an SLD model with only faulty nodes detection

is better adapted, since the virtual links are not uniquely implemented, and they may change in time.

Both the adaptive and one-step approaches could be used, but due to the way one agent should test another (see Section 4) the one-step method was found to be less time and bandwidth consuming.

The implemented algorithm is based on ideas from [7]. Chutani and Nussbaumer propose an adaptive algorithm, where tests are performed depending on previous results. We used the same algorithm, but with an underlying test graph built as described in [20], which is not necessarily complete. Such a test graph was proven [20] to be sufficient to diagnose a system said "one-step diagnosable". In the DIANA project, the system is diagnosable if at least half of the number of agents is reliable; if more than this number are faulty, the system is considered to be undiagnosable and the SLD algorithm will not be able to find which are the faulty and fault-free agents.

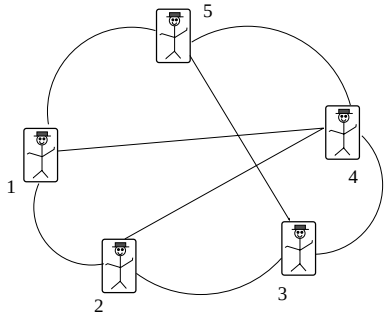

Figure 1: Test graph for five agents

Such a hypothesis does not seem to be unreal, since it is very unlikely that a system is half unreliable. If this is the case, something is actually going very wrong, and a message like "System undiagnosable" should be sufficient as a result to warn the network manager.

Using the hypothesis that at most half the number of agents is faulty, the algorithm tries to find one fault-free agent, and using the results of the tests performed by this agent (and the other fault-free diagnosed agents) all the system is diagnosed.

The algorithm is the following:

Input: Test graph S, set of nodes $V(S)$,

Output: a fault-free unit $u_i \in V(S)$, or an undiagnosable system.

1. Initialize the state, $U := V(S), T_S := \emptyset$, $Candidates := \emptyset$,
2. While Length$(U) > 0$ and Length$(Candidates) < \frac{|V(S)|+1}{2}$ do
3. if (Length $(Candidates) = 0$), select an arbitrary node $u \in U$, put u at the end of the list $Candidates$ and set $U := U \setminus \{u\}$

4. If Length(U) > 0, let u_j be the last element of *Candidates*. Find a node $u_i \in U$ such that the result of u_i testing u_j is fault-free;
   - if there is such a node u_i, place u_i at the end of *Candidates*, remove u_i from U, and goto step 2,
   - otherwise, remove u_j from the end of *Candidates*, goto to step 2.
5. If Length (*Candidates*) $\geq \frac{|V(S)|+1}{2}$, the first element in the list is good; otherwise the system is not diagnosable.

In the next sections we describe what a fault-free (or reliable) agent is, and how the test will be performed.

3 Reliable Agents

An agent is considered to be **reliable** if the data that it is supposed to possess is reliable, i.e., correct and updated. This is meant to guarantee that an agent correctly perceives its domain, where **domain** here means the set of network elements the agent is responsible for, and that its data-base matches reality.

A great part of the data stored by an agent comes through the network, sent by an SNMP, CMIP or other agent. Some of the information is polled continuously and some is requested when necessary. As most of the action an agent has to perform is based on all this information (comparison with thresholds, state changes, etc), freshness must be a property required for the collected data.

So, an agent will be said to be reliable not only if it possesses correct information reflecting the state of its domain in the network, but also if its information is recent. This property would imply that the agent has no problems in accessing the network and the network elements, and in processing and storing the information coming through the network.

We face now the problem of finding out a way to verify that a domain is correctly perceived without checking the agent whole database and comparing it to the data coming from the network elements. Some representative elements must be chosen in the agent domain, and may include servers, printers, routers, switches, hosts, etc.

Of course the choice of the representative elements depends strongly on the definition of the agent's domain. For instance, if the domain of an agent consists of elements in a same Ethernet line section, then checking a sample containing some elements of this subnetwork may suffice to guarantee that what the agent sees is correct. But if the agent's domain consists of several routers of a network, or elements in different sections, then the route from this agent to one element may be (most probably is) different from the one

to another element in a different sub-network. In this case, choosing only some of the elements of the domain may not be sufficient.

4 Testing

One of the most important points in the SLD implementation is the choice of the test. From a theoretical point of view several tests exist: symmetrical, asymmetrical, and others (see [3] for a review), but in practice the test depends on the system to be diagnosed, on the properties and characteristics of the system and its components, on the available testing tools, and this may even include the protocols available when telecommunications systems are considered. In this section we propose a procedure for an intelligent agent to test another one, using the idea of representative elements, considering that the system is composed of intelligent agents designed for network management.

As said before, each agent is in charge of monitoring and storing information of a certain set of network elements, the agent's **domain**. In order to diagnose one agent, its neighbors will check if the information that this agent possesses are correct and updated.

In order to do this, to each agent will be assigned a test domain. The **test domain** of an agent consists of its own representative elements and the representative elements of its neighbors (see Fig. 2). Independent of the information needed for the normal talks of the agent, a minimum set of information must be stored for the elements in the test domain. This set of information must be the same for all agents and all representative elements, so that information may be easily exchanged and compared.

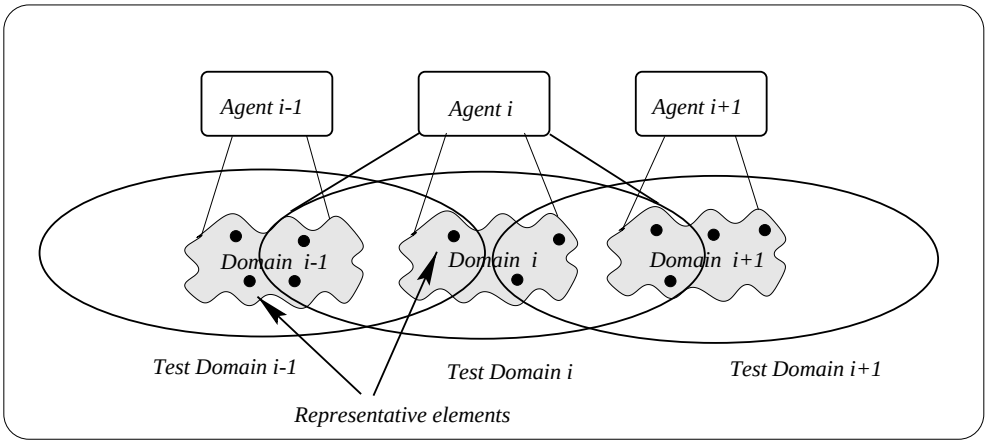

Figure 2: Test domain

For instance, the set of important information for one representative element may be composed of:

- the response to a Ping request,
- the network element sysUpTime of the last status request,
- the status of the first interface in the interface MIB table.

The agent database for diagnosis purposes would then contain a table as depicted in Table 1, where each element E_i is a representative element in the agent test domain.

Network Element	Ping?	sysUpTime	interface status
E_1	OK	...	up
E_2	OK	...	down
E_3	OK	...	up
E_4	NO	?	?
...	...	...	...

Table 1: Test domain table

The test procedure is the following: periodically each agent sends to its neighbors the information it has collected of these neighbor representative elements, and also receives from each neighbor information about the neighbor representative elements. The received information is then compared to the data locally collected. The test consists in checking, for each neighbor and each representative element, that variables like Ping response or interface status have identical values in both samples, and checking that the values of a variable as sysUpTime differ in the two samples only in a reasonable amount of time – the acceptance interval. If a certain percentage of the representative elements of one neighbor passes the test, then the agent considers this neighbor to be reliable. This percentage may vary from network to network; in the DIANA demonstration the absolute majority, (i.e. $> 50\%$) was chosen as the number of elements that should succeed the test.

The periodicity of the test and the acceptance interval depend strongly upon the polling frequency (and this depends on the number of representative elements being monitored). Experiments must be made to reach a balance between these values, so that wrong conclusions are not produced because of too small and/or large values. Of course, a less frequent periodicity implies having a greater acceptance interval, but other factors may influence the final values.

As it was defined, the test is clearly asymmetrical, since each agent tests the information concerning the representative elements of its neighbors, an not its own. So, since different tests are being performed, the outcome of A testing B and B testing A may be different. This can occur when different sets of representative elements are reached differently by different agents. In Section 6 we point out how this kind of representative elements information can lead to some conclusions about the network reachability.

5 Implementation

In the DIANA agent architecture [9], the SLD is implemented as a *skill*, that is, an independent module that is loaded only when needed, and that consists of several tasks whose goals are specialized in a domain activity. Two different skill modules are necessary: one common for every agent, the SLD or slave module, and one special for the master agent, the SLDM or master module.

The agent that is asked to load the SLDM module is considered to be the master, and should perform some special functions besides the simple SLD test. The result about the agents state is also updated by the master agent, and is available to the other skill modules.

Master role

1. Get the agents list (pre-requisite information)
2. Create an information per agent, representing the agent status
3. Determine each agent neighborhood
4. Ask the other agents to start the SLD, sending to them the needed data for the SLD module to start
5. Repeat
   (a) Wait for the slave agents to send their diagnosis, and consolidate the result
   (b) Updates each agent status

Slave role

The slave starts with the information given by the master: the master identification and its neighbors. In order to execute, this module needs that a monitor module updates information about the representative elements. To consider the diversity of domains, the agent may be told which kind of representative elements and how many elements must be chosen. The agent role is then the following:

1. Filter the agent domain to obtain the representative elements, using the type of each network element
2. Send to its neighbors its representative elements
3. Get its neighbors representative elements

4. Ask to the monitoring skill module to update the beliefs about its own representative elements, and the representative elements of each neighbor agent
5. Repeat
   (a) Get the local monitoring result
   (b) Send to the neighbors the local monitoring result
   (c) Get the distant monitoring result
   (d) Consolidate the results obtaining the diagnosis about its neighbors
   (e) Send to the master its diagnosis about its neighbors
   (f) Wait until next consolidation time

In the demonstration, the goal is to have information about some network elements (NE), and the agents monitor the NE in a delegated way. The information collected by the agents is sent to a master agent; this one, based on the diagnosis given by the SLD, decides to use or not the collected information, and to redistribute the domains, if the status of an agent changes. A scenario showing the utility of SLD was prepared, where the fault of a NE is not noticed, because the agent that should monitor this NE was not reliable when the fault occurred.

6 Further Improvement in Diagnosis

With the information collected for the agent system diagnosis, more than a simple comparison can be done. Inferences about the information concerning several elements can be drawn, if sufficient data about the topology of the network is maintained.

What will be described in this section are examples of how one could diagnose more than only the intelligent agent system; diagnose further means to extend the procedure of testing an agent to a more comprehensive understanding of the network state.

For instance, if a set of elements in a same subnetwork do not answer to a Ping request, then there must be a network problem to attain this subnetwork, and the agent cannot conclude anything about this domain. If only some elements do not answer, then these network elements may be down. In this case if the used protocol is unreliable, before warning the manager, the agent should repeat the request.

For each representative element, if one of the collected sysUpTime is too old, then most probably there is a problem with the status request or with the network element, and a new request should be done (if the protocol is unreliable). But if this happens for a great number (to be defined) of network elements, then the agent that has the oldest

data has some problem in treating the informations it receives and will be seen as faulty by its neighbors; remark that this agent will see its neighbors fault-free.

The agent should now compare the network element "status" of the elements that have an updated information. If they differ for one element, then another request may be made. And, as before, if this happens for a great number (to be defined) of network elements, then one of the agents has a problem. In fact in this case each agent will see its neighbor faulty.

Table 2 summarizes the situation. The column *Correlation* means that the elements that present the same variable values are reached using the same route starting from the agent. A white cell implies that the associated value is not relevant for the diagnosis, or is not available. The notation $t_2 >> t_1$ means that t_2 is more recent that t_1, and $t_2 \simeq t_1$ means that the values are within an acceptable interval.

Agent_1			Agent_2				
Ping	*Time*	if *Status*	*Ping*	*Time*	if *Status*	*Correlation*	*Result*
No						No	NE problem
No						Yes	Reachability problem
OK	t_1		OK	$t_2 >> t_1$		No	NE or SNMP problem
OK	t_1		OK	$t_2 >> t_1$		Yes	Agent_1 unreliable
OK	t_1	S_1	OK	$t_2 \simeq t_1$	$S_2 \neq S_1$	No	NE or SNMP problem
OK	t_1	S_1	OK	$t_2 \simeq t_1$	$S_2 \neq S_1$	Yes	Agent_1 or Agent_2 unreliable

Table 2: Diagnosis table

7 Conclusion

The Intelligent Agent technology is being more and more used in treating all kind of information; in the case of network management, where machines monitor machines, a special care must be taken with respect to the reliability of the information.

In this paper we discussed how to ensure reliability using the System Level Diagnosis. A diagnosis test specially designed for the network management using a multi-agent system was presented, and we showed how the SLD technique was implemented in the scope of the DIANA project.

As already pointed out, the information collected for the diagnosis test may be also used to have an overall idea of the network health state and its connectivity. Some work in this direction will be conducted within the DIANA project, where the main idea is summarized be the scheme depicted in figure 3.

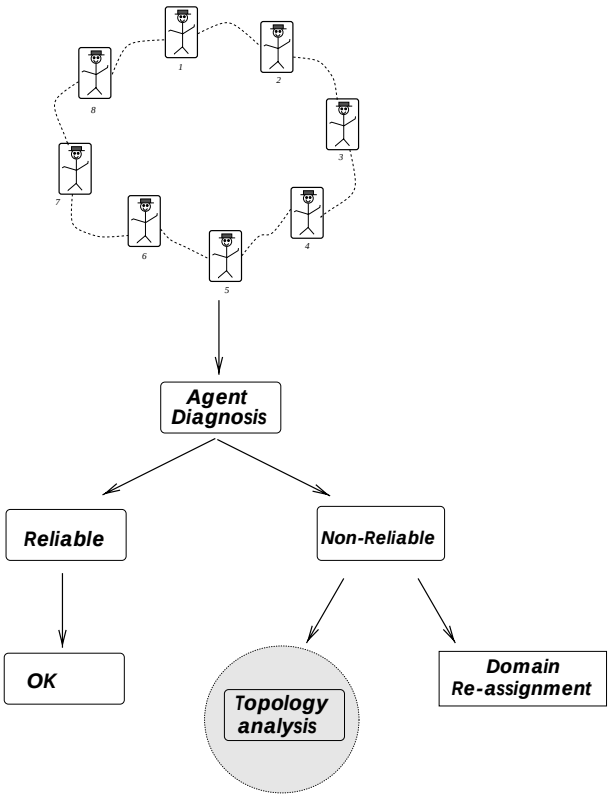

Figure 3: Future work

The SLD technique can be also used and adapted to other specialized multi-agent systems. The most important point – as with every SLD application – is to design a suitable diagnosis test, as the one presented in this paper.

Acknowledgments

Thanks to all the DIANA team for their support and useful discussions that allowed to achieve this work: P. Conti, M. Cheikrouhou, and J. Labetoulle.

References

[1] A. K. Bansal, K. Ramohanarao, and A. Rao. Distributed storage of replicated beliefs to facilitate recovery of distributed intelligent agents. In *Intelligent Agents IV*, Lecture Notes in Artificial Intelligence, pages 77–91. Springer Verlag, 1997.

[2] M. Bearden and R. P. Bianchini Jr. Efficient and fault-tolerant distributed host monitoring using system-level diagnosis. In A. Schill, C. Mittasch, O. Spaniol, and C. Popien, editors, *Distributed Platforms*, pages 159–172. Chapman & Hall, Feb. 1996.

[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] R. P. Bianchini Jr. and R. W. Bunskens. Implementation of on-line distributed system-level diagnosis theory. *IEEE Transactions on Computers*, 41(5):616–626, May 1992.

[5] A. T. Bouloutas, S. B. Calo, A. Finkel, and I. Katzela. Distributed fault identification in telecommunication networks. *Journal of Network and Systems Management*, 3(3):295–312, 1995. Distributed Fault Management, Management Domain, Alarms, Alarm Domain.

[6] S. Brugoni, G. Bruno, R. Manione, E. Montariolo, E. Paschetta, and L. Sisto. An expert system for real time fault diagnosis of the italian communication networks. In *Proceedings of the IFIP TC6/WG6.6 Third International Symposium on Integrated Network Management*, pages 617–628. Elsevier Science Publishers, 1993.

[7] S. Chutani and H. J. Nussbaumer. Extending the theory of system-level diagnosis for communication networks. Technical Report 94/58, École Politechnique Fédérale de Lausanne, Département d'Informatique, Lausanne, CH, Aug. 1994.

[8] S. Chutani and H. J. Nussbaumer. On the distributed fault diagnosis of computer networks. Technical Report 94/56, École Polytechnique Fédérale de Lausanne, Lausanne, CH, 1994.

[9] P. Conti, J. Labetoulle, K. Marcus, and M. Cheikhrouhou. Network management system with intelligent agents. a first step with sld. In *HPOVUA'98 Workshop*, Rennes, FR, 1998.

[10] R. Cronk, P. Callahan, and I. Bernstein. Rule-based expert systems for network management and operations: An introduction. *IEEE Network*, pages 7–21, 1988.

[11] R. Davis and W. Hamscher. Model-based reasoning: Troubleshooting. In W. Hamscher, L. Console, and J. de Kleer, editors, *Readings in Model-Based Diagnosis*, pages 3–24. Morgan Kaufmann, 1992.

[12] R. Deng, A. Lazar, and W. Wang. A probabilistic approach to fault diagnosis in linear lightwave networks. In *Proceedings of the IFIP TC6/WG6.6 Third International Symposium on Integrated Network Management*, pages 697–708. Elsevier Science Publishers, 1993.

[13] E. P. Duarte Jr, T. Nanya, G. Mansfiel, and S. Noguchi. Non-broadcast network fault-monitoring based on system-level diagnosis. In A. A. Lazar, R. Saracco, and R. Stadler, editors, *Integrated Network Management V*, pages 597–609, San Diego, USA, May 1997. IFIP, Chapman & Hall.

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

[15] N. R. Jennings and M. Wooldridge. Software agents. *IEEE Review*, pages 17–20, 1996.

[16] G. A. Kaminka and M. Tambe. Social comparison for failure detection and recovery. In *Intelligent Agents IV*, Lecture Notes in Artificial Intelligence, pages 127–141. Springer Verlag, 1997.

[17] S. Kätker and K. Geihs. A generic model for fault isolation in integrated management systems. *Journal of Network and System Management*, 5(2):109–130, Feb. 1997.

[18] L. Lewis. A case-based reasoning approach to the resolution of faults in communication networks. In *Proceedings of the IFIP TC6/WG6.6 Third International Syspposium on Integrated Network Management*, pages 671–682. Elsevier Science Publishers, 1993.

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

[20] 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.

[21] S. Rangarajan, A. T. Dahbura, and E. A. Ziegler. A distributed system-level diagnosis algorithm for arbitrary network topologies. *IEEE Transactions on Computers*, 44(2):312–334, Feb. 1995.

[22] 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.

[23] F. Somers. HYBRID: Unifying centralised and distributed network management using intelligent agents. In *Networks Operation and Maintenance Symposium (NOMS96)*, 1996.

[24] 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.

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