Multi-Agent Systems and Network Management - A Positive Experience on Unix Environments

Nelson dos Santos Júnior[1,2], Flávio Miguel Varejão[2] e Orivaldo de Lira Tavares[2]

[1] Companhia Siderúrgica de Tubarão
Av. Brigadeiro Eduardo Gomes, s/n - Jardim Limoeiro
CEP 29164-280 - Serra - ES - Brasil
nsantos@tubarao.com.br

[2] Universidade Federal do Espírito Santo - UFES
Centro Tecnológico - Programa de Pós-Graduação em Informática
Av. Fernando Ferrari, s/n - Campus de Goiabeiras
CEP 29060-900 - Vitória - ES – Brasil
{tavares, fvarejao}@inf.ufes.br

Abstract. This paper presents an experiment of using a multi-agent system that improves the efficiency in network management on Unix environments. This system aims to decrease the financial damages occasioned by processing interruptions of the computational environment. The multi-agent system is based on a distributed and delegated management approach and it was developed using the GAIA methodology for analysis and project of agent-based systems. The agents architecture is based on activities and rules base. The agents have lists of activities and apply their rules bases to accomplish them.

Keywords: multi-agent system, network management, unix, intelligent agent, gaia methodology.

1 Introduction

Currently, the computational environment of many companies is composed by a great variety of hardware components, software, operating systems, etc. Depending directly on their computational environment, there are several applications (systems) that companies use to support their main businesses. The stability and readiness of this computational environment have become more and more important so as to allow greater productivity and efficiency of companies. Slowness or even unavailability of some of the company most critical applications could be the result of any unstable or unavailable part of the computational environment, and can generate large financial damages [10] [12].

This article presents a work on how the multi-agent system-based technology [1] [3] can increase the readiness of computational environments. The multi-agent system has been gradually developed to deal with the different parts of computer network management. Initially, the experiments have been only performed on Unix environments. On these environments, a common problem that often causes damages to the stability of user applications is the *file system full* problem. We have chosen this problem to observe the multi-agent system performance. It happens when some disk space shared by many applications is 100% used. Then, some applications become unavailable and others have their response time significantly enlarged. The traditional way of handling this problem depends on human experts. Monitoring tools or complaining users notify the experts about the problem and they must identify the problem causes and take the corrective actions. The main pitfalls associated to this procedure are the possible unavailability of human experts (the problem may happen at any time, including during the night, weekends and holidays) and delays on the problem solving (the problem causes and their corrective actions may not be trivial). It is important to emphasize that delays on the file system unavailability increase the possibility of provoking a cascading effect over the overall network environment, suspending and aborting many other applications.

The multi-agent system was developed using the GAIA Methodology [18] for analysis and project of agent-based systems, and aiming the implementation of a multi-agent system in the Companhia Siderúrgica de Tubarão (CST) [3]. Due to the very fact that it is a metallurgical company producing 24 hours a day, the main businesses in CST are supported by applications which must be maintained stable and available as long as possible. The multi-agent system is able to identify the causes of the file system full problem and take actions such as canceling processes and removing or reallocating files, therefore, generating new free disk spaces and increasing the environment availability. The multi-agent system is composed by several specialized agents acting on different parts of the Unix environment. Each agent may react to a list of hierarchically ordered activities and have a knowledge base describing how to perform these activities.

Analysis of the log of our experiment has shown that the multi-agent system avoided that important financial and production control systems were interrupted and provided disk space for executing routine backup procedures (which, otherwise, would not be performed).

This article has the following structure: section 2 introduces a general view of the area of network management and services. Section 3 presents a description of the multi-agent system. In section 4, we describe the experiment being performed. Finally, in section 5, the conclusions and future works are presented.

[3] http://www.cst.com.br

2 Network Management Approaches

The network management main approach has been the IETF Management Structure (SNMP). The SNMP structure is based on a small group of concepts: agents or managed nodes (which represent elements of the network, as bridges, routers, switches, etc.), manager or network management system (which provides a set of operations to the network support team), managed information base - MIB (represents the information that can be manipulated by each agent), the management protocol - SNMP (standard way of extracting management information from the systems) and the proxy (allows the interaction of the SNMP structure with components that don't include this structure) [10] [12].

The SNMP structure is a centralized approach which uses distributed agents to collect manageable information. However, this approach has some problems when dealing with extensive and complex network models. The central manager may have some troubles for handling great amounts of manageable information and for managing a large number of geographically distributed sites [12] [16].

An alternative approach uses the "divide and conquer" strategy for promoting solutions based on distributed management and on delegation [12]. Several works use this approach, such as proposed by BRITES [2], GOLDSZMIDT [7] and OLIVEIRA [14]. In this approach, a main manager delegates the control to several distributed workstations and to more powerful agents. By promoting the collaboration among these agents, we have several management "islands". This procedure increases the reliability and failure tolerance of the network environment [12].

Another approach, considered by many researchers as the ideal solution for the management complex network management, is the paradigm of mobile agents' [4] [10]. The term Mobile Agent refers to autonomous programs which can move through the network, from node to node, and assume the knowledge of an agent, i.e., acting as users or other entities [15]. Researches have been made in distributed network management and delegation [7], network services deliver [9], optimization of network traffic and failure tolerance of networks [11].

Whichever approach is adopted, it should aim to promote the pro-active concept in the network management. Several approaches have used this concept, such as, CHIU[5], DE FRANCESCHI [6] and HOOD [8].

3 The Multi-Agent System

The multi-agent approach is based on distributed management and on delegation. The agents are permanently acting in specific parts of the computational environment, being able to solve from simple routine problems (of their strict domain) to even more complex problems that involve communications and activities among several agents. Our approach was chosen based on the characteristics of the problem to be solved (the need for immediate actions instead of having to wait for human intervention), as well as on the intelligent agents characteristics (autonomy to take actions, always active and ready to take actions, capacity to act together with other agents, speed equivalent to the computational processors in which they are installed, etc.) [17].

The multi-agent system was developed using the GAIA[4] methodology [18] for analysis and project of systems based on agents. The multi-agent system includes the following roles: *ADMINISTRATOR*, *MONITOR*, *COORDINATOR* and *CONTROLLERS*.

The computational environment represented in figure 1 is composed of several hardware components which host multi-agent systems. Inside these hardware components, there are software components and other components that the system may control. The monitor role is also part of the computational environment and is represented by the monitoring tools, as shown by the "black box" in figure 1.

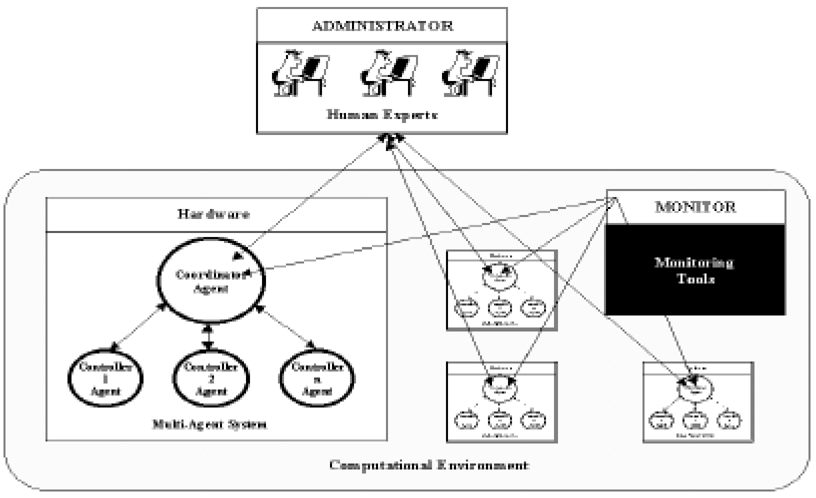

Figure 1 - Global Architecture

Each multi-agent system has an agent that coordinates the other controlling agents. These agents are responsible for the communications among the controlling agents and also allow the interaction between the multi-agent system and the human experts. In addition, they receive the information about the problems detected by the monitoring tools and distribute them to the appropriate controlling agents.

The controlling agents are responsible for the specific control of services and parts of the computational environment. For instance, we may have safety controllers, database controllers, response time, disk space, cleaning, printing service, etc.

[4] http://www.inf.ufes.br/~tavares/dsba

Research with similar architectures are found in the works of ABECK [1] and MARZO [13].

All agents have the same architecture. This feature allows the construction of a unique agent programming structure. The agent architecture is based on activities and rule bases. The agents should perform activities and apply their rule bases to accomplish them. The distinction between agents is expressed in their rule bases. Figure 2 shows that, if there is no longer any activity to be accomplished, the agents change their state to “sleeping”. This state is disabled as soon as a new activity to be accomplished is sent to the agent or the sleeping time ceases.

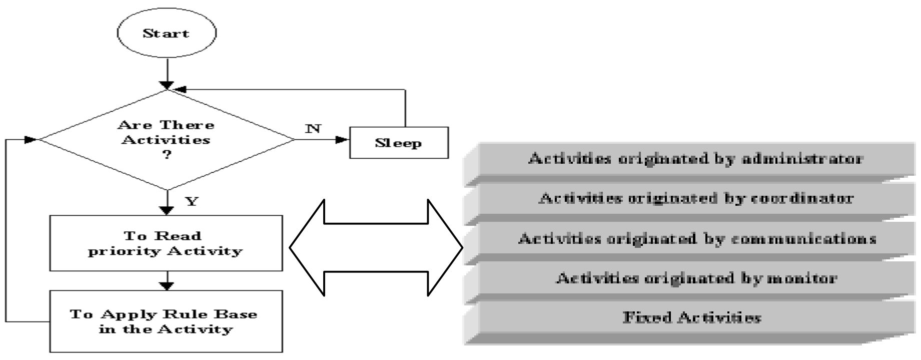
Start
Are There Activities ?
N
Sleep
Y
To Read priority Activity
To Apply Rule Base in the Activity
Activities originated by administrator
Activities originated by coordinator
Activities originated by communications
Activities originated by monitor
Fixed Activities

Figure 2 - Agent Processing Architecture

Every agent selects the highest priority activity to perform first, as viewed in Figure 2. The fixed activities are executed after the “sleeping” state. The priorities sequence guarantees the compliance with the proposed hierarchical structure because an agent will always accomplish an activity generated by a human expert first, no matter how many fixed or communications activities it must have to accomplish in a given moment.

We developed a *coordinator*, a *disk space controller*, *a processes controller*, and a *cleaning controller agent* for solving this problem. The disk space controlling agent has the function of controlling the allocation of the computer disk space in order to maintain the environment stability. The processes controlling agent is responsible for controlling all the processes executed in the computer, analyzing CPU consumption, looping states, process ownership and the interdependencies among the processes. The cleaning controlling agent is responsible for controlling and optimizing the use of files inside the computer. It controls where the main files are and who is responsible for them. It also performs the cleaning of temporary files.

4 The Experiment

The strategy to obtain results in the experiment was divided in choosing an appropriate environment for the multi-agent system's operation, to choose a problem with appropriate complexity degree, to test the agents individually, to test the agents' united operation with those changing communications and finally to analyze the results obtained with the multi-agent system's operation in a period of three months.

4.1 The Environment's Choice

The experiment development strategy consisted of choosing a part of the computational environment that was critical in terms of availability, involving a high level of complexity and demanding non trivial solutions and the participation of several controlling agents acting together to solve them. The Unix environment was the one chosen. The main applications assisting the metallurgical processes of CST use databases installed in Unix servers. The prototype was initially installed on eight Unix servers: two of development, one of endorsement and five of production (financial, supply, histories, control and supervision). The language chosen for the development of the agents was the language Korn Shell because it is native of the Unix environment (thus, computationally efficient) and allows the complete environment control.

4.2 The Problem's Choice

The *file system* problem usually happens when a Unix environment process saves (or is still saving) large files or a large number of small files in the disk area. This problem solution consists of canceling the process creating the files, whenever it is still active, and then removing or moving those files to some free disk area in the same computer or in another one. That problem allows a good observation of the multi-agent system's operation, because it demands the constant communication among the agents to be solved. Another important factor for the experiment, is of the problem to happen sometimes a week and in any schedule, making possible real verification of the multi-agent system's operation. Finally, the causes and solutions of the healthy problem very well defined, as well as the flow of communications and activities that the agents would have to execute.

4.3 The Individual Tests of the Agents

The first step for the acceptance and trust in the multi-agent system is it of trusting that the agents individually work. For that programs were prepared to simulate a problem in a specific area of the environment, where it knew each other the causes and improve solutions. After the release of the problem, each agent was activated and soon after having disabled following the necessary flow to solve the problem, and they were made analyses in the logs of those agents' performance. This phase demanded plenty tests and corrections in the knowledge bases agents' .

4.4 The Multi-Agent System's Test

After each agent's individual operation to be approved, the multi-agent system was completely activated, with all the agents working at the same time, but prepared to solve and to investigate only the specific problem. Parallel to the multi-agent system's operation, several programs were activated to take care of the safety for unexpected effects of actions of the system. Those programs identified relative abnormalities to parameters of the Unix environment that could be prejudiced for the multi-agent system, and if that it happened, it would disable the same. The next phase was it of to throw the problem and to collect the agents' logs and to validate the performances. A following important phase was the stress test of the system, where the problem was thrown at random and automatically about 30 times a day for 10 days. After the good results of the stress test, the system was prepared to investigate any problems in the environment, but only informing that actions would have taken. After 30 days with appropriate operation, the knowledge bases of the agents were fed with information from where it could take actions and where it would not be able to act. Finally the multi-agent system was send for the other Unix servers.

4.5 Detailed operation of the Multi-Agent System

In general lines, after being notified by the monitoring tools of some problem in some area of the environment, the multi-agent system has to reproduce actions that specialists would take to solve that. Basically, it would be to identify which the files that are in the moment the responsible for the increase in the occupation rate and processes are existed using those files. Identified the processes and files, it is canceled the processes and the area is liberated removing or moving the files.

Initially, the monitoring tool sends an activity for the coordinating agent, whose content informs where it is the area with problems and which the current rate of occupation. In the experiment, the monitoring tool was configured to verify all the areas of the computer every 3 minutes and to generate a problem for the multi-agent system, if some rate it was above 95% of occupation.

The coordinating agent verifies for the type of the activity which would be the responsible controlling agent for solving the problem and directs it for that agent. In that case, the space disk controlling agent is the responsible and receives the activity directed by the coordinating agent. In the experiment, the coordinating agent's knowledge base was fed with the responsible by resolutions of several types of problems and the result was I always need.

The space disk controlling agent, when receiving the activity, looks for initially to confirm if the problem continues to happen. Confirming that, it tries to identify the causes of the problem. Identifying the causes, it directs extra activities of cancellation of processes and cleaning of files for the coordinating agent, that has the role to distributing the activities. In the experiment, the causes were precisely identified. Those varied of problem for problem, being able to be sometimes originated by an or more files with an or more processes. Problems found in the identification they were current of lack of rights for investigation in certain directories or due to the problem already to have finished when the agent began the identification. In that case the agent the problem concluded with status of anything to do.

The coordinating agent again received activities, this time originated by the space disk controlling agent and identified which agents would be the responsible for those, directing to proceed.

The processes controlling agent and the cleaning controlling agent had in your knowledge bases, relationships of countermands for cancellation certain processes as well as removal certain files. In the experiment, when those countermands were identified the specific agent it could not solve the activity and it returned that information for the coordinating agent that directs the impossibility for the space disk controlling agent. The space disk controlling agent in that case, documented the impossibility of solving the problem and that concluded, informing the specialists the found countermands. Not having involved countermands, the processes controlling agent would usually cancel the requested processes and the cleaning controlling agent would remove the suitable files. In the experiment, we chose until we consolidate the acceptance of the system, for the agents to inform the specialists that them agents would have done to solve the problem, as well as informing to the responsible for the affected systems of those actions. The analysis of the logs of the actions taken up to now by the agents they were shown very positive, not having indication of disastrous actions for the environment.

Finally the controlling agents came back for the coordinating agent the execution of the activities that directs those information for the space disk controlling agent that the problem concluded.

4.6 Obtained Results

In the table 1, we see a summary of the obtained results.

Table 1: Results Obtained in Multi-agent System's Experiment

Time of Operation of the Experiment	90 days – Feb until May of the 2002
Amount of Total Problems	78
Amount of Resolved Problems	68
Percentile of Resolved Problems	88%
Benefits reached by the system	Avoided that important financial and production control systems were interrupted; Provided disk space for executing routine backup procedures; Avoiding problems of response time for maintaining critical file systems below 100% of occupation.

The reasons of not resolution of problems for the multi-agent system was identified with lack of rights of investigation certain directories and problem already concluded in the moment of the analysis. The results obtained with the experiment of a multi-agent system's operation it has been very positive, generating some acquired important knowledge besides in the experiment, such as in the identification of responsible places for most of the problems and the specialists' performance close to the responsible for the definitive solutions as well as in the popularization of technology use based on Artificial Intelligence for the organization with practical experiment bringing trust in your use.

5 Conclusions and Future Work

This work shows how an Artificial Intelligence based technology, not frequently used in the routine working environment of companies, can be of great value for keeping available the company computational environment.

The multi-agent system may avoid to stop parts of the industrial production, failures in performing financial operations that may provoke fines, human expert time spent solving complex problems that began simple, loss of time from users, customers having their purchases impeded, accidents with our without human loss, etc. The good results obtained by the multi-agent system have motivated claims from the human experts to include new and more complex problems to be solved. New knowledge is being included to the agent rule bases and new agents are being constructed to attend these demands.

In the short term, we intend to add more controlling agents to the Unix environment to evaluate the growing difficulties of an increasingly complex agent structure acting together. The next controllers will be the database ones (a type of omnipresent DBA), the safety ones (to control the access to the computers, invasions attempts, password control, etc.), the response time (to maintain the environment with suitable performance rates) and other specific services controllers. It is also our intention to develop in the medium term agents using a language compatible with other environments that can be controlled (NT, Netware, WEB, etc.). In the long term, we intend to endow the agents with a type of learning capacity, through the development of "curious" or "questioning" agents which register situations in which the multi-agent systems have failed. These agents will keep questioning the human experts about how these agents were solved. Another type of learning happens when the agents themselves inform the agents of the same type located in other hardware about alterations in their knowledge base. With the growing complexity of the knowledge bases, it would be ideal to have a way to make the agents themselves optimize those bases in order to get better solutions than the ones previously used. In this context, the agents will be able to teach the human experts.

References

1. ABECK, S.; KOPPEL, A.; SEITZ, J.; A Management Architecture for Multi-Agent Systems. Proceedings of the IEEE Third International Workshop on System Management. Newport, Rhode Island. Apr., 1998.

2. BRITES, A.; A Protocol-Independent Notation for the Specification of Operations and Management Applications. Proceedings DSOM'94, Fifth IFIP/IEEE International Workshop on Distributed Systems: Operations & Management. Oct. 1994.
3. BRITO, S. R.; TAVARES, O. L.; Agentes Inteligentes: Definições, Aplicações e Comunicação. Revista Engenharia - Ciência & Tecnologia. ISSN 1414-8692. Vitória-ES, Ano 2, No. 9, p. 13-20; mar.-abr. 1999.
4. BUCHANAN, W.J.; NAYLOR, M.; SCOTT, A.V.; Enhancing Network Management using Mobile Agents. Proceedings of the 7th IEEE International Conference and Workshop on the Engineering of Computer Based Systems. Edinburgh, Scotland. Apr. 2000.
5. CHIU, T.; Getting Proactive Network Management from Reactive Network Management Tools. International Journal of Networks Management. Vol. 8, Issue 1, Pag. 12-17, 1998.
6. DE FRANCESCHI, A.S.M.; ROCHA, M.A.; WEBER, H.L.; WESTPHALL, C.B.; Proactive Network Management Using Remote Monitoring and Artificial Intelligence Techniques. Proceedings of the 2nd IEEE Symposium on Computers and Communications (ISCC'97). Jul. 1997.
7. GOLDSZMIDT, G.; YEMINI, Y.; Delegated Agents for Network Management. IEEE Communications Magazine. Vol. 36 Num. 3, Pag.: 66-71, Mar. 1998.
8. HOOD, C.S.; JI, C.; Intelligent Agents for Proactive Fault Detection. IEEE Internet Computing. Vol. 2, No. 2, Mar./Apr. 1998.
9. KRAUSE, S.; MAGEDANZ, T.; Mobile Service Agents enabling Intelligence on Demand in Telecommunications. Proceedings of IEEE GLOBCOM'96, 1996.
10. LIPPERTS, S.; Enabling Alarm Correlation for a Mobile Agent Based System and Network Management - A Wrapper Concept. Proceedings of the IEEE International Conference on Networks. Brisbane, Australia. Oct., 1999.
11. LOPES, R.P.; OLIVEIRA, J.L.; Software Agents in Network Management. Proceedings of the 1st International Conference on Enterprise Information Systems - ICEIS'99. Setúbal, Portugal. Mar. 1999.
12. LOPES, R.P.; OLIVEIRA, J.L.; On the Use of Mobility in Distributed Network Management. Proceedings of the 33rd Hawaii International Conference on System Sciences. Maui, Hawaii, Jan., 2000.
13. MARZO, J-L.; VILÁ, P.; FABREGAT, R.; ATM Network Management based on a Distributed Artificial Intelligence Architecture. Proceedings of the fourth International Conference on Autonomous Agents. Pag: 171-172. Barcelona, Spain. Jun. 2000.
14. OLIVEIRA, J.; Arquitectura para Desenvolvimento e Integração de Aplicações de Gestão. Tese de Doutorado. Universidade de Aveiro. Sep. 1995.
15. PHAM, V; KARMOUCH, A., Mobile Software Agents: An Overview. IEEE Communications, Vol. 36, Num. 7, pag: 26-37, Jul. 1998.
16. SPRENKELS, R.; MARTIN-FLATIN, J-P.; Bulk Transfer of MIB Data, The Simple Times, Vol. 7, Num. 1, Mar., 1999.
17. WOOLDRIDGE, M.; JENNINGS, N.R. Applications of the Intelligent Agents. In: N. R. JENNINGS, N. R.; WOOLDRIDGE, M. (Org.). Agent Technology: Foundations, Applications and Markets. New York: [s.n.], 1998.
18. WOOLDRIDGE, M.; JENNINGS, N.R.; KINNY, O. A Methodology for Agent-Oriented Analysis and Design. In: O. Etzioni, J.P. Muller, and J.Bradshaw, Editors: Agents'99: Proceedings of the Third Annual Conference on Autonomous Agents, Pag: 69-76, Seattle, WA, 1999.