45
Proactive Management of Computer Networks using Artificial Intelligence Agents and Techniques

Marco Antonio da Rocha
Federal University of Rio Grande do Sul (UFRGS)
National Supercomputing Center and Computer Science Institute (CESUP)
91520-130, Porto Alegre, RS, Brazil
Fone/FAX: +55.51.339-46-99, E-mail: rock2@cesup.ufrgs.br

Carlos Becker Westphall
Federal University of Santa Catarina (UFSC)
Network and Management Laboratory (LRG)
Caixa Postal 476 - Campus Trindade
88040-970, Florianópolis, SC, Brazil
Fone: +55.48.231-97-39, FAX: +55.48.231-97-70, E-mail: westphal@lrg.ufsc.br

Abstract

This work was developed in the area of Computer Network Management. The work is intended to establish a strategy for the implementation of proactive management in the available management environment, i.e. the National Supercomputing Center, for the management of networks associated with the use of agents. The work was motivated by a need to explore the use of agents to identify symptoms of proactive management problems which might occur in networks, and especially to recognize a problem using artificial intelligence techniques and take reactive measures to solve it, configuring a proactive management application for the prevention of problems in computer networks.

Keywords

Proactive management, baseline, artificial intelligence, rules of production, management networks

1 Introduction

Due to the constant interaction between different types of computer network users, the need is increasingly felt for better organizational techniques in order to manage the resources offered. There no longer exists a single user profile, standard solutions are inadequate, and it is now necessary to divide efforts and apply diversified solutions which attend to the interests of each work group which makes up the network. Network management is a distributed application which involves the exchange of information among the management processes, with the goal of monitoring and controlling diverse network resources. The processes involved with a specific area assume two important roles: Manager and/or Agent. The Manager is part of a distributed application which generates operations and receives reports. The Agent is part of a distributed application which generates objects associated with it (responding to requests for operations from the Manager and giving reports which reflect the operation of the objects). Admitting that the management tools don't encompass the wide range of network problems and that they aren't always applied by network operators, it becomes necessary to apply other management mechanisms in order to overcome the most apparent shortcomings.

A. A. Lazar et al. (eds.), *Integrated Network Management V*

Using this as a starting point, a need was felt to cover this particular subset of computer networks that are not attended to by currently available management tools and which address the individual differences of each organization which separate it from the others, as well as making it easier to administrate. Thus, it becomes possible to have two possible operations in a computer network: Reactive Operation, in which the problems are reported to the manager for action, and Proactive Operation, in which the management should be capable of detecting problems, and avoiding them, before they occur.

Therefore, this paper seeks to present an adopted strategy for the implementation of proactive management in computer networks. The work that motivated the elaboration of this report was driven toward the implementation of an agent and the joint utilization of network monitoring and commands of the UNIX operating system for the management of computer networks, where the equipment makes up the network of the Institute of Computer Sciences and of the National Supercomputing Center, including a SunNet Manager platform, Sun workstations, Sun OS 4.1 operating system compatible with UNIX 4.3 BSD, Sun OS C compiler, Open Windows 2.1 and PCs in a network configuration with the National Supercomputing Center were used.

This paper is organized in the following manner: in Section 2, the relative aspects of network management in general are addressed, a proactive management scheme is presented, as well as observations on how artificial intelligence techniques can be applied in this paradigm; in Section 3, a detailed treatment is presented of a strategy for implementation of a minimum proactive management prototype, as well as a rule set for use in detecting network problem symptoms; in Section 4, a commentary is made as to the results obtained; finally, in Section 5, the final conclusions are presented.

2 Network Management and Proactive Management

The flow of information in a network should be reliable and rapid, which implies that the data undergo constant monitoring, so as to filter or even detect problems which can result in losses. A network can exist without management mechanisms, although its users might encounter difficulties with congestion, security, routing, etc. Management is oriented towards controlling activities and monitoring network resource use. Simplified, the basic duties of network management are: obtain information from the network, treat these data for possible diagnosis and execute solutions to the problems. To reach these objectives, the management functions should be contained in diverse components of the network, allowing for the discovery of, prevention of and reaction to problems [WES 91, WES96].

To solve the problems associated with network management, the ISO, by OSI/NM, proposes three models: the Organizational Model, which establishes a hierarchy among management systems within one management environment, dividing the environment to be managed into various domains; the Informational Model, which defines the objects of management, their interrelationships and the operations made upon these objects. A MIB is needed to store managed objects; the Functional Model, which describes the functions - error, configuration, performance, account and security - of management. In this way, the concept of network management provides the administrator with sufficient means by which to distribute the network's resources to its users, while, due to the quantity of information available, allowing for proactive management.

2.1 Proactive Management

As previously stated, the concept of proactive management involves the anticipation of possible problems which may occur in a computer network and their detection before they occur, and not merely reporting their existence. It is fundamental that abnormal network operation be observed, that symptoms be collected and that larger problems which may come to occur be diagnosed correctly, or that anomalies be registered when it is not possible to collect enough evidence which associates an event with a known problem. It is also necessary to maintain constant observation of the network, so that based on this knowledge enough data can be collected in order to identify what might be a symptom and relate it to a known problem. In an ideal situation, the LAN management tools establish a framework

within which devices like smart hubs can monitor network activity and place information at distributed management platforms which automatically will generate trouble ticket, operational costs and usage reports [JAN 93].

These same platforms should provide services through the application of management with the capacity of configuration and planning. In Proactive Management, it is necessary to have a comparison of many management tools so that, when data is computed, a report can be created which indicates the cause to be inspected. The majority of these tools work with thresholds which establish the limits where the reporting of events like number of errors, specific types of packets and other parameters regarding the selection of intervals is to begin. It is also important to add related databases which makes for easier access and more versatile use of management data, making the integration of other tools possible.

Through the measurement of normal activity within a given period of time and the identification of performance based on statistical calculations, it is possible to establish a body of data with normal function parameters which we call the baseline. This baseline can be used in Proactive Management by a set of functions to establish a valid statistic which characterizes the normal operation of the network during a new period of time for a specific interval, assessing the levels of traffic at different times on different days [JAN 93].

2.2 Artificial Intelligence and Proactive Management

As we have seen earlier, network management is a complex job, in which we find support for other areas of applications, among which Artificial Intelligence and expert systems are especially prominent. The principal management products already use these to facilitate management functions. Artificial Intelligence can be used to anticipate problems that would leave the network inoperative. In this way, adopting Artificial Intelligence can contribute to Proactive Management. System monitoring can be used to project the network's performance, comparing data taken with a baseline which takes into account the proper choice of corrective action. The job of interpreting and diagnosing a network malfunction is a strong point in expert systems, which can rely on inferences regarding the collected data. Design and planning of new installations can be performed with a good knowledge based installed.

The use of expert systems has recently increased, principally in those areas where complex functions are performed but where few specialists exist. Computer networks fit this context due to their wide dispersion and growing use. Services offered by a network become indispensable; many people depend on them so much that an investment in their management becomes viable. A quick cost-benefit survey shows the following advantages: better quality of service: with the dissemination of the specialist throughout all segments of the network. The administrator's job is facilitated, resulting in better performance; greater agility, lower costs and greater productivity in the execution of services permitted by automation; higher reliability, with decreased decision-making time; training support for improved human resources preparation.

A expert system is divided into four distinct phases: acquisition of knowledge; knowledge base; inference machine; explanatory interface. The acquisition of knowledge is a phase involving extraction and formulation of knowledge from an expert for use in a expert system. In this process, work is performed with "knowledge engineers", technicians specialized in the job of helping experts put their knowledge into the expert system using practical rules and knowledge structuring. As the expert puts forth his or her knowledge, the knowledge engineer represents it as a set of heuristic rules which, when coded, drive the process by a mass of information, making the process more efficient. Thus, obtaining these rules is an important step in the acquisition of knowledge.

The knowledge base stores the knowledge of the expert and differentiates from a conventional database in that it is active in nature, permitting updates conforming to the context. The structure of the knowledge base will depend on the type of knowledge represented. To have deductive knowledge, the base will usually be composed of rules. To have modeling of physical structures, causal links or interrelationship between models, the ideal structure may be a semantic network.

The inference machine selects and applies the appropriate rule during each step in the expert system, manipulating the knowledge base. The inference machine can base itself on premises or elementary bits of information, and tries to achieve its objective through a combination of the two. In

this case, it is said that it finds its way forward. The machine can also base itself on an objective and verify the needed premises using the facts involved and arrive at a conclusion. In this case, it is said that it works backwards. Inference machines that use a mixture of these approaches are those which are most successful, since, in most cases, the choices made in the inference process are reproductions of the processes a human would be likely to employ.

In a expert system, the knowledge of the problem's domain is organized separately from the other system knowledge, such as the procedures or steps for problem solving, or interaction with the user represented by the explanatory interface, which defines how to present the knowledge. This division is intentional because these systems divide themselves according to knowledge base (the store of specialized knowledge) and inference machine (which unites the procedures for fixing problems, or steps for solving them). The combination forms what is called a knowledge-based system. The base contains facts and rules, and the inference machine decides how to apply these rules and in which order so as to obtain new knowledge. Once the specialized knowledge is separated, it becomes easier for the designer to manipulate procedures [ROC 94a].

3 Implementing a Minimum Proactive Prototype

In the following section, the implementation of a minimum proactive prototype is presented for the purposes of making a practical validation of the work. Considering what has been stated until now, the objective of proactive management is to develop a proactive approach in one of the functions of the functional management model, with performance management being studied as a consequence of the desire to foresee drops in network performance and at the same time anticipate the events which may come to clog the network or result in greater damage. It should not only advise of the occurrence, but also take measure to avoid letting such problems expand and become critical.

To reach this goal, a method was idealized where the process remains active, periodically monitoring remote systems and analyzing the results along with other information taken from the baseline, where the manager receives traps from agents, and which plays a role in the diagnosis of network problems, taking action according to threshold levels or merely communicating a fact which has occurred. In short, proactive operation. [ROC 94a]

3.1 Proactive Management Applied to a Performance Analysis of a Network Gateway Machine

During this article, individual details of the presently achieved workflow will be presented, showing the adopted strategy for its elaboration, but abstracting specific details such as the SunNet Manager interface and the method of developing an agent for the management environment [SUN 89]. The task of monitoring and observing the network is best achieved by agents with resident action in the machines to be managed. In view of this, Agent 6 was created with the intent of measuring quality and operation in the network communication services by monitoring the volume of traffic and verifying the number of errors. These characteristics will later be used as a database for establishing a baseline.

Having these data, the following workflow was determined: run agent 6, Hostmem, Hostif and Hostperf in some sub-network nodes, measuring congestion and other statistics in different hours; based on the results, establish a baseline or base with known points as a measure of standard deviation for measurements in normal situations; implant the diagnostic model, so as to activate rules each time the measurement taken is beyond the standard parameters contained in the baseline, stipulated as: for each measurement taken from the monitoring agents that arrives within reach of the timed measurement in which it occurred, a diagnostic module will be triggered to verify whether of not problems exist according to the module's rules; if the diagnostic module verifies the a problem exists which could result in a performance drop or congestion, rules will be used to determine the motives for the event; in a third moment, after verifying the previous, measures are taken to avoid the problem, reporting the anomalies found to the network administrator and suggesting corrective measures.

3.2 Environment of the Experiment and Strategies Used

In this experiment, the local network at CESUP (the National Center for Supercomputing) was used, which represents a highly heterogeneous network. At CESUP, IBM-PC compatibles, Apple Macintosh computers, Silicon Graphics workstations, Suns and a Cray YMP2E supercomputer are employed, and more specifically, the SUN workstation sub-network.

The Macintosh computers communicate to each other using the AppleTalk protocol. The physical connection is achieved via a hub, which is the star configuration in this sub-network. The AppleTalk sub-network connection with the CESUP local network is by a gateway.

The 143.54.22 sub-network is an Ethernet segment, and there are found the Silicon Graphics and Sun stations. Another Ethernet segment is sub-network 143.54.1, which is the backbone of UFRGS, where the Tchepoa router (port of entry to the Tche network), Vortex (name server of the university network) and Routcc (router for the Downtown Campus). Remote access to CESUP, for users from other institutions, is done via Tchepoa and Vortex. Access by UFRGS users is via Routcc.

Uniting these two sub-networks, we find the Darwin station, which plays a key role in the CESUP local network. This machine is the only access node to the Gauss supercomputer (CRAY YMP2E) for both local and remote users. Connection to Darwin and the Gauss is via proprietary CRAY protocol. Darwin also serves the local network archives, e-mail, names (DNS) and yellow pages (NIS).

3.3 Agent 6

To obtain a practical validation of this experiment, the problems of performance and congestion were also chosen, since the results in those areas would serve as feedback for other studies. The objective was to find a mid-region for performance drop at a given workstation and when levels near this region were reached, send a message to the network administrator alerting the beginning of a drop in performance and congestion. The term congestion is defined here as the point where many packets are present in a branch of the sub-network which cause a degradation in performance, i.e. when the number of packages sent does not correspond with the number of packets distributed. Congestion differs from traffic volume primarily in the distribution of transmissions. If the volume is high, but is being managed by just one machine, then the network congestion, from the point of view of that machine, is not considered to be high.

As cited in [ROC 94], the average used for measuring network congestion is based on a comparison of the number of attempts made at transmission by the interface of one host of an Ethernet channel with the number of collisions which occurred in this transmission. If the percentage of collisions is high, this indicates that the network is congested, for there is little free time without someone transmitting. This measurement obviously depends on the machine in question.

If the machine in which the agent is running is generating high traffic and the rest of the machines on the network are generating almost nothing, then the number of collisions will be low, not reflecting the true volume of transmission achieved. The true volume of traffic can only be measured by counting all of the packets that pass through the network. Meanwhile, this verified difference will not adversely affect the average desired congestion, since if the station at which the agent is running is transmitting too much, it does not perceive congestion. And for the rest of the stations, there will only be congestion when they try to transmit more, which will also be reflected at the agent's station by an increase in the number of collisions. To obtain a percentage of collisions, the relation between the number of collisions and the number of attempts at transmission which occurred during the interval between the present time and the most recent consultation [ROC 94].

Agent 6 was installed in the Darwin workstation, which is the server for the local network and the gateway for CESUP, being monitored the ie0-bus interfaces of the UFRGS network and the ie1-bus of the CESUP local network. Agent 6 operated in two basic modes supported by the SunNet Manager (SNM) platform: data monitoring and event monitoring.

For data monitoring, the agent remained in continuous execution, making a consultation at each time interval. At the end of each consultation, the data recovered was sent to the SNM manager. This periodic sending of data permits the generation of graphs (see figure 1) by the SNM platform, as well as

the storage of recovered data from previous consultation by the same agent, which can then generate statistics about the status of the communication system. Since the SNM platform supports the automatic generation of events like "send mail" and permits the specification of various types of thresholds, the implementation of this kind of operation by Agent 6 brought no burden to its development. To the contrary, this facility was used in sending alerts to the network administrator.

Simple monitoring of data at each small interval in time does not permit easy conclusions as to whether or not the network is seriously congested or not, since flurries of transmission happen normally. For example, if it were possible to obtain the proposed average at each attempt at transmission, the data recovered would always indicate 0 or 100% congestion, which would be difficult for one to interpret. This situation is aggravated in the event monitoring mode, since it does not interest the network administrator to receive an event informing that the rate of collisions was, for example, 30% during the last second. On the other hand, if a measure of congestion is obtained during the entire life of the system, the average would be quite low and would not adequately reflect the problems which had occurred during determined periods.

In order to allow for more intelligent control of a congestion situation on the network, it is necessary for the agent to give an average reading over the last N time intervals. The choice of this number N should be based on modeling the system's situation adequately in relation to real time and according to the administrative needs. From this average, we can then have a wider vantage point and truly manage useful events, informing that network congestion is on the rise and is surpassing the limits imposed for correct operation.

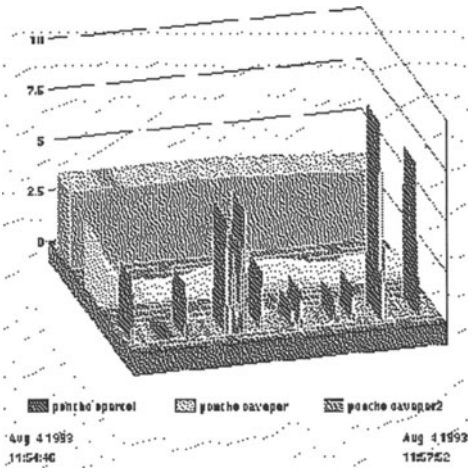

Figure 1: Graphic with the Attributes of Agent 6 in a Real Situation

The implemented agent possesses unique table called "output". In this table, all the parameters corresponding to the table of the like-named etherif and three more attributes specific to Agent 6 are recovered. These attributes are "opercol" -- percent of collisions at each basic interval; "oaveper" -- average percent over the last 60 intervals; and "oaveper2" -- average percent over the last 900 intervals.

The calculation of the averages is updated at each interval. This calculation does not consume much CPU time because the algorithm used maintains the sum of the last 900 and 60 intervals in memory, allowing it to merely subtract the oldest average, add the newest, and make two divisions (60 and 900) to obtain the latest averages. The calculation of the average at each interval (each consultation) is performed by dividing the difference in collisions by the number of attempts at transmission in a period. This number of attempts at transmission is calculated as the sum of the difference of attempts at retransmission and the difference of packets transmitted successfully. The average calculated over 60 basic intervals exhibits successive elevated peaks which appear as reasonably large increases in

transmission congestion, indicating that the situation is critical in that period. The average calculated over 900 basic intervals exhibits low sensitivity to alterations caused by any application requiring large transfers. To the contrary, its high stability better reflects the state of utilization of the network in general, for all users and during a longer period of time (above 15 minutes).

3.4 Building a Baseline

This station was continuously monitored by Agent 6 for 24 hour periods, taking measurements at 10 and 15 minute intervals, seeking the best interval of measurements per hour, with a total of 4 to 6 measurements per hour for a daily total of 96 to 144 measurements. After making the measurements, evaluations were made regarding the progress and profile of the traffic measured during this period and the objects which best contributed to the establishment of reference levels of operation were selected, respectively, for Hostperf, Hostmem, Hostif and Agent 6. These are: Hostperf => cpu - percent of CPU utilization; ipkts - number of incoming packets; opkts - number of outgoing packets; ierrs - number of incoming errors; oerrs - number of outgoing errors; colls - number of collisions; Hostmem => mbuf - percent of buffers used; memused - bytes used by the network; memfree - bytes free to the network; mem - percent of bytes used by the network; Hostif => ipkts, opkts, ierrs, oerrs and colls - same meaning for ie0 and iel interface; Agent 6 => oaveper, opercol, oaveper2 - mentioned in the previous section; odrops - cumulative number of packets in the output queue; obuff - current number of buffers transmitting;

Each agent collects data from the gateway machine and generates ASCII files which contain the values encountered in each parameter that they must measure. The objects selected are extracted from these archives via a "parser" program which is specific for each agent, written in C language, which collects them in an intermediate file, calculating the average and the standard deviation of the values of the object for each hour. This intermediate file is passed by another general parser program, which also calculates the average and the standard deviation of the values of each object, uniting each monitoring day within its respective weekday, thus obtaining the average standard deviation for each hour of each day throughout the period monitored. With this baseline, we have an average standard operation for the network for each hour of each day of the week.

3.5 The Diagnostic Module

The diagnostic module was elaborated with all the characteristics of a expert system, utilizing the four phases of this type of system which are presented here. Methodologies proposed by [LEA 95] for the construction of knowledge trees were employed in the process of knowledge acquisition. To complement this, a bibliographic review of available material was performed and soon after the first attempts at modeling the system (represented using a Generic Semantic Model, or GSM), were made. In the figure 2, the final modeling of the system is presented.

The knowledge tree construction methodology, also proposed by [LEA 95], associates diagnostics with their determining factors, which are proposed according to importance. By this process, a set of trees was built which represents the knowledge base for the system. From these trees the rules composing the rule base of the diagnostic module were derived, and it was identified that the diagnostics are realized on four level. In Figure 3, three of them can be seen: the lowest is the parameter level, where the state of each parameter is identified, being that it is evaluated as a function of its value as well as of the average and the corresponding deviation at present days and hours; the partial diagnostic level contains diagnostics which are made through parameter analysis, though they are not important for the user, and; the upper level of final diagnostics, which are presented to the user. Also, at the level of the tree appearing in the diagram, the suggestion level was applied, where, as a function of the final diagnostics, suggestions are made to the network administrator.

Knowledge is represented by facts and rules. The facts are generated from the monitoring and baseline files. The implementation of this system was achieved in Prolog. The files that arrive via the system have their format converted to the Prolog fact form. The facts have the following format: agent

(name-attribute, value); previous (name-attribute, value); agent (interface, name-attribute, value); previous (interface, name-attribute, value).

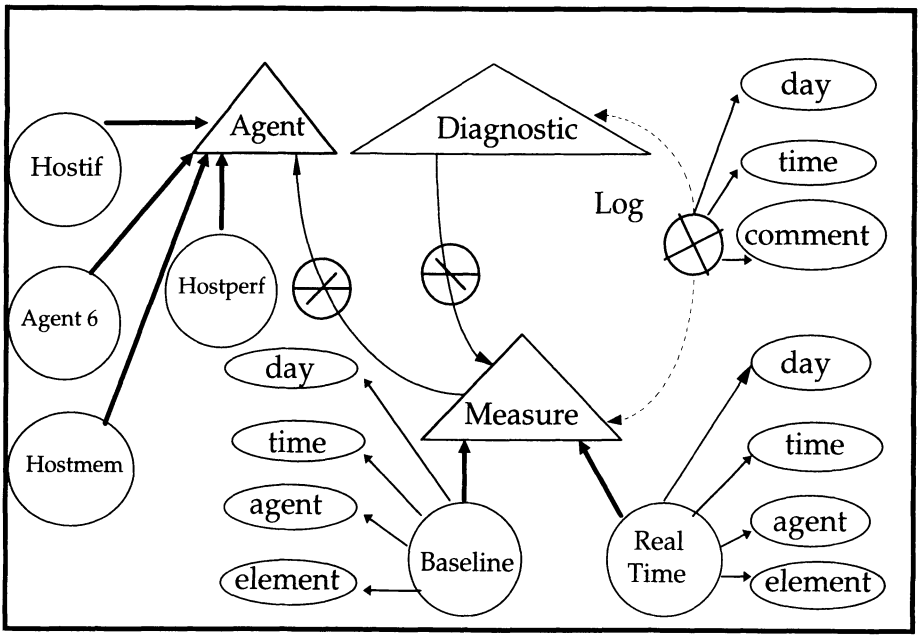

Figure 2: Modeling using GSM

The hostif agents and Agent 6 depend on the interfaces of each machine, and for this reason they also save this value. The previous value of each parameter should be saved in the case of cumulative measurements, and only the difference between the previous and present ones are important.

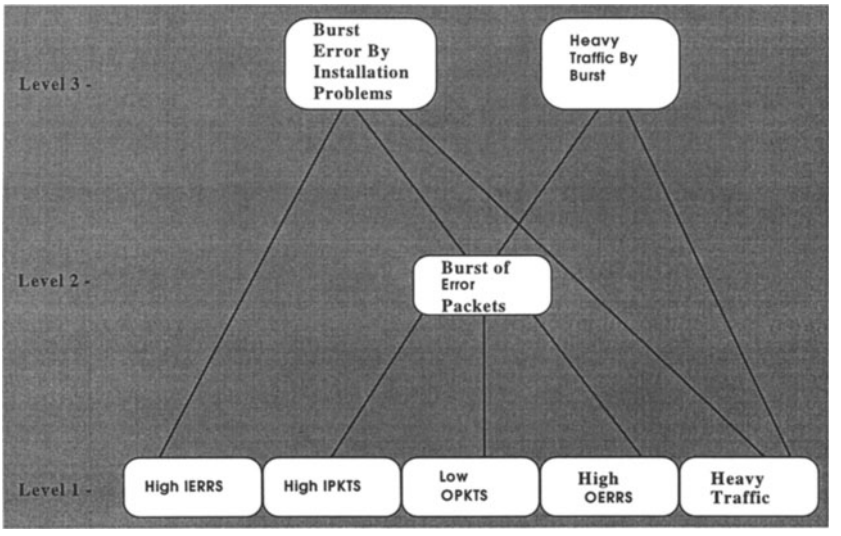

Figure 3: Partial Representation of the Knowledge Tree

As the prototype was implemented in Prolog, operators were used for defining a syntax for rules which represent the knowledge of domain. Having defined these operators, the rules can be written in a more accessible and adequate syntax by the specialist, and included in the interpreter to facilitate the implementation of the inference machine: theory "Teoria" (Theory); rule "N" (N); if "ListaDeCondições" (ListOfConditions); then "Consequência" (Consequence); because "Explanação (Explanation). Where "Teoria" - identify the ruleset. The inference machine can be triggered for just one set of rules, instead of surveying the entire rule base; "ListaDeCondições" - a conjunctive list, in Prolog format, of conditions; "Consequência"- routine Prolog call; "Explanação" - translation of the rule in natural language, used as an option for explanation.

```
teory final
rule 6 :
if
      [parametros('CPU high'),
      parametros('OPERCOL high'),
      parametros('OAVEPER high'),
      parametros('OAVEPER2 high'),
      parametros('IPKTS high'),
      parametros('OERRS low')]
then
      goal(final('Low performance'))
because
'Was detect use of CPU over normal and parameters OAVEPER e OAVEPER2 greater than
zero, with more packets in and out of machine grather than normal. This machine is
loaded because have high number of colisions and rate of error'.
```

Figure 4: Low Performance Rule

Figure 4 illustrates an example of a rule used in the system, with the above-mentioned syntax: The first set of rules serves as a parameter which identifies situations which are abnormal with respect to the parameters given by the agents. Therefore, the value obtained in the monitoring file generated by the agents is compared to the average baseline value, taking into account that the baseline represents a "normal" network standard, per hour and day of the week. The second set is called partial and detects alert situations that should be investigated by the system, but not presented to the user.

The final set takes into account the parameter and final diagnostics to identify situations which should be presented to the network administrator. The rules in the suggestion set take into account the final diagnostics and supply suggestions to the administrator for avoidance or elimination of the problem. Mapping between knowledge tree levels and the rule sets is direct. The organization of the rules base in "rulesets" makes it easier to implement the inference machine, as well as making the search more efficient. The inference machine was implemented upon the resolution mechanism of Prolog itself, upon which the prototype was implemented. Given a set of rules, the mechanism takes each rule of that set and tests its premises, backtracking. The Prolog algorithm is showed in figure 5.

The algorithm treats a rule, with or without an explanation (because), tests the conditions set represented by a list, executes the rule action, marks the region triggered [goal(marca-trace(Teoria, N)]. The register of rules that have been triggered is later used by the explanation mechanism. After a rule is tested, a failure is forced so that, by backtracking, the rest of the regions can be tested. In the prototype, this inference mechanism is invoked for each rule set, sequentially. This highly procedural strategy makes inference easier to implement.

The explanatory interface seeks the diagnostics at the top of the tree that have been detected, sets up the path that led to each diagnosis and a text using the "because" sentences from each region triggered on the path. The following is a representation of this algorithm: assume T as the theory set in the rule base; "t" is the top theory in the theory hierarchy T; identify the set R of rules which pertain to the theory t which have been triggered; for each rule r of R; 1. set up the path C of rules triggered that led to trigger of r; 2. follow the path C, from top to base of knowledge tree, treating "because" sentences.

```
diag(Teoria):-
      (teory Teoria rule N : if LCond then Cons;
      teory Teoria rule N : if LCond then Cons because _),
      testa_cond(LCond),
      call(Cons),
      goal(marca_trace(Teoria,N)),
      fail.
diag(_).

testa_cond([]).
testa_cond([C|R]):-
      call(C),
      testa_cond(R).
```

Figure 5: Prolog Algorithm for Inference Machine

The explanation mechanism sets up the entire explanation of all paths at once, responding because it was given a suggestion by the proactive management system, because each final diagnosis was given, and so on. Thus, the model has conversion blocks; knowledge base; inference engine; explanation; interface.

The conversion block is responsible for receiving the monitoring and baseline files and converting them to the Prolog fact format. These facts will form the knowledge base. Based on the knowledge base, the inference engine is used for analysis of the values of the parameters and for inferring a diagnosis. This diagnosis is supported by an explanation which indicates the motive(s) for the problem, as well as suggesting possible resolutions for it. From the interface, the network administrator receives information from the system as well as the suggestions and has the possibility of expressing an opinion, agreeing or not with the diagnosis given, Beyond this, it should, in whatever situation, describe which of the approaches were followed in trying to solve the problem.

4 Results

For security reasons, the present prototype runs on an IBM/PC compatible microcomputer connected to the network, and Arity Prolog was implemented. For testing, the monitoring files generated by the Darwin machine agents had to be copied directly to the prototype directory, being that, during this test period, only one type of problem was verified.

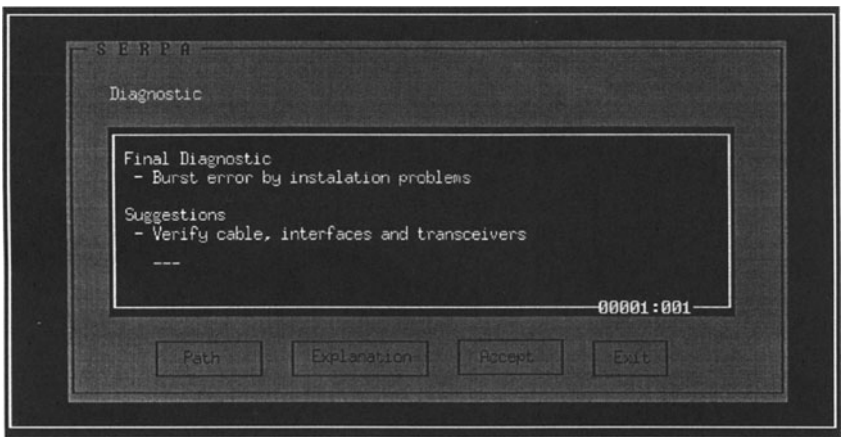

Figure 6: Diagnostic Demonstration Screen

Once executed, the prototype automatically converted the monitoring data and consulted the baseline statistics, using the hour and date of the system as a base, it informed the user of the final diagnosis and of suggested actions to be taken by the network administrator, as shoed in figure 6.

From here, it was possible to request an explanation the system interface is comprised of: Central Window, which is used to show diagnostics; Path Option, a simplified explanation, showing which rules were triggered, but without assembling them; Explanation Option, explains the rules used by the system; Accept Option, registers the file in the log; Exit Option, to leave the system.

If the explanation has a number of lines above that available on the monitor, exceeding the window's size, as happened in the example, it can be scrolled down.

The Accept Option registers those suggestions which have been accepted by the administrator in a log file; if he or she leaves the system without accepting any, this fact is also registered in the log file. The purpose of the log is to validate the system itself, making it possible to later investigate the situations in which the administrator did not accept the suggestions, so as to create a record which can be used by the inference machine to construct diagnostics. The automation of this process is very important, because the daily work routine of the network administrator normally does not have time to manually procure a log file, principally due to the size of these files.

The prototype proved its utility, since the symptom "flurry of error packets" enabled the administrator to take rapid measures, due to the fact that notification of the event was received before the event was completed.

5 Conclusion

With the objective of validating the use of proactive management in computer networks, this study proposed a strategy for the implementation of proactive management using artificial intelligence, more specifically, systems based on knowledge. The area of network management chosen for this experiment was that of operation management, analyzing the problem of performance drops in a "gateway" machine (network server), observing how its operation influenced the network.

Theoretical and practical activities were performed to verify that agents were indeed viable for network monitoring and, with this data in hand, it was possible to construct a database containing the normal state of the network. Thus, it was possible to verify the operation of the "gateway" machine (network server). To this end, the following steps were performed: Agent 6 was developed to be responsible for monitoring the network and analyzing the performance of the Ethernet channel; "parser" programs were developed in C language to place monitoring files in the baseline file component format; a baseline was assembled with the Agent 6 results plus the three proprietary Hostmem, Hostperf and Hostif agents; a diagnostic module was elaborated with characteristics of a expert system to inquire as to the existence of performance problems, and if confirmed, establish the motives which caused the drop in performance. Once executed, the prototype automatically converted monitoring data and consulted the statistical information of the baseline, taking the system date and time as a base, informed the user of the final diagnosis and offered suggested actions to be taken by the network administrator.

The prototype proved its usefulness, in that the "flurry of error packets" diagnosis, given its subtle characteristics, would otherwise demand much time to detect, while using the tool enabled the administrator take rapid measures, due to the fact that notification of the event was received before the event was completed. It would be interesting to focus on this work and more fully research ways to amplify the diagnostic module and perfect this prototype. In addition, we can cite the following items to be improved: to implement automatic interconnection of the diagnostic module and the monitoring files; to improve explanation interface to emit suggestions to users by e-mail automatically, and; to implement other areas of network management, such as configurations and failures.

In the way of new tendencies, we can cite the following items which we believe to be a natural sequence of this work: to extend the work to other technologies like FDDI, ATM, etc.; to develop the diagnostic module with a coexisting system; to use the diagnostic module on natural networks; to develop proactive management utilizing a combination of artificial intelligence with simulation.

The problem of network management is far more complex and requires careful treatment in order not to create new problems when trying to solve others. This holds even more true for performance management, traditionally treated separately. In proactive management, every care should be taken to anticipate problem occurrences and to not induce administrators to seek solutions for virtual situations. This article, besides presenting an implementing strategy, showed results from the intended model which served to demonstrate that the operation can be attained. The experiment demonstrated that CESUP possesses a well-constructed network, with stable operation and performance, which did not allow us a greater number of problems to be inspected, though the problem which did occur was detected and reported to the administrator, confirming proactive management's capabilities. Its operation and principals can be used as a base for new works to be performed on the proposed model.

REFERENCES

[COM 91] COMER, D. E. Internetworking with TCP/IP: Principles, Protocols, and Architeture. Volume 1. Seg. Edição. Prentice-Hall, Englewood Cliffs, NJ, 1991.

[DEF 96] DE FRANCESCHI, A.S.M.; KORMANN, L.F.; WESTPHALL, C.B. Performance Evaluation for Proactive Network Management. In: Proceedings of IEEE/ICC'96 International Conference on Communications, Dallas, Texas, USA, Jun. 23-27, 1996.

[HUL 87] HULL, R., KING, Roger. Semantic Database Modeling: Survey Applications and Research Issues. ACM Computing Surveys. v.19, n.3 Setp., 1987.

[JAN 93] JANDER, Mary. Proactive LAN Management: Tools that Look for Trouble to Keep LANs Out of Danger. Data Communications. Mar., 1993.

[LEA 95] LEÃO, Beatriz. Class notes from COMPO2 course CPGCC/UFRGS. Porto Alegre, 1995.

[POS 81] POSTEL, J.B. Internet Protocol. Request for Comments 791, DDN Network Information Center, SRI International, Setembro, 1981, 45pp.

[POS 81a] POSTEL, J.B. "Internet Control Message Protocol - DARPA Internet Program Specification". Request for Comments 792, 1981.

[ROC 94a] ROCHA, M.A. A Strategy for Implementation of Proactive Management in Computer Networks. Research Report, Porto Alegre, UFRGS-CPGCC, Mar. 05, 1994.

[ROC 94] ROCHA, M.A. Computer Network Management through New Agents. Proceedings of the XII Brazilian Symposium on Computer Networks, p.113-133, Curitiba, PR, Brasil 1994.

[STE 90] STEVENS, W.R. UNIX Network Programming.Prentice-Hall Inc. Englewood Cliffs, NJ, 1990.

[SUN 89] Sun Microsystem Inc. SunNet Manager Tutorial - How Write an Agent, 1989.

[SUN 89a] Sun Microsystem Inc. Network Programming Guide, 1989.

[SUN 90] Sun Microsystems Inc. SunOS Reference Manual. Vol I, 1990.

[WES 91] WESTPHALL, C. B. Conception et développement de l'architecture d'administration d'un réseau métropolitain. Thèse de Doctorat nouveau régime. Université Paul Sabatier. Toulouse, 16 Juillet 1991.

[WES 96] WESTPHALL, C. B. & KORMANN, L.F. Usage of the TMN Concepts for Configuration Management of ATM Network. International Symposium on Advanced Imaging and Network Technologies. Germany, Berlin Oct. 7-11, 1996.

ABOUT THE AUTHORS

Marco Antonio Rocha

Obtained his B.Sc. degree in Informatics at PUC - Pontifícia Universidade Católica - Rio Grande do Sul, Brazil, in 1989 and a M.Sc. degree in Computer Science at Federal University of Rio Grande do Sul in 1996. At present, he is lecturer at the University of ULBRA in the south of Brazil and also works at CRT which is the Brazilian Telecom carrier placed in Rio Grande do Sul.

Carlos Becker Westphall

Obtained a degree in Eletrical Engineering in 1985 and a M.Sc degree in 1988, both at the Federal University of Rio Grande do Sul, Brazil, and a Dr. degree in Informatics - Network Management - at the Université Paul Sebatier, France, in 1991. Presently, he is a Professor in the Department of Computer Science at the Federal University of Santa Catarina - Brazil, where he acts as the leader of the Network and Management Laboratory and also coordinates the multi-institutional PLAGERE project (Platforms for Network Management) funded by the Brazilian National Research Council (CNPq).