NMESys: AN EXPERT SYSTEM FOR NETWORK FAULT DETECTION

Peter C. Nelson and Janet Warpinski

Department of Electrical Engineering and Computer Science
University of Illinois at Chicago
Chicago, IL 60680

Abstract. The problem of network management is becoming an increasingly difficult and challenging task. It is very common today to find heterogeneous networks consisting of many different types of computers, operating systems, and protocols. The complexity of implementing a network with this many components is difficult enough, while the maintenance of such a network is an even larger problem. This paper presents a prototype network management expert system, NMESys (pronounced nemesis), implemented in CLIPS. NMESys concentrates on solving some of the critical problems encountered in managing a large network. The major goal of NMESys is to provide a network operator with an expert system tool to quickly and accurately detect hard failures, potential failures, and to minimize or eliminate user down time in a large network.

1.0 INTRODUCTION

The problem of network management is becoming an increasingly difficult and challenging task. Networks can fail at many different components, connections and levels, often potentially disrupting service to many users. Sometimes, portions of the network can detect a failure in other portions of the network. Other times, the fault may go undetected by the network. NMESys (Network Management Expert System) is a prototype network management system which monitors alarms in a network and helps a network operator determine points of failure. NMESys is able to receive and decipher information from components in the network about detected failures. NMESys also has the ability to proactively interrogate the network to find undetected failures. The primary goal of NMESys is to detect and isolate faults so they can be repaired with minimal or no user down time.

2.0 SYSTEM OVERVIEW

There are many problems to be solved when managing a large heterogeneous network. First, just determining that a failure has occurred can be a difficult problem, even if the failing component or an adjacent component has detected the failure. Often large networks post many types of messages to report conditions which may or may not warrant operator intervention. Operators may be bogged down researching dead ends, while some major component is in a state of failure. NMESys is able to filter out the messages which do not indicate hard failures. This allows operators to concentrate on the problems which pose the greatest threat to the integrity of the network.

Another problem in managing the network is determining conditions which are degrading the network. Often error messages do not necessarily indicate a hard failure, but rather some temporary error condition. Since the component recovers, these conditions often go unattended. With many of these types of messages being generated in a large network, operators do not have the time to research and resolve each one. However, this type of event can be an indicator of a more severe problem which is starting to manifest itself. If the problem goes on long enough, it may result in an extended outage of a component which could have been repaired before the hard failure occurred. NMESys tracks all conditions which have been reported to it, even if the indication was non-fatal. For degrading conditions, threshholds are utilized to determine when a non-fatal condition warrants attention. If a threshhold has been exceeded within a certain time frame, then an error message is posted to the network operator indicating that a potentially serious condition has developed. Thus the problem can be addressed before the failure even occurs.

NMESys always knows the status of each component in the network, since it receives all messages which indicate changes in state of the components. The system can always give an operator the current status of any component or set of components. This can be very helpful to an operator to know how many components are down at any one time. NMESys also has the ability to show the message history for any component so that problems can be researched. In addition, NMESys can give an operator a list of events which have occurred in the network which need action. As operators take care of these events, they are acknowledged. This allows the system to always have a current list of events warranting attention.

NMESys has the ability to interrogate components in the network about their current status. This functionality provides a proactive approach to detect conditions which, for whatever reason, have not been reported properly. Often, error conditions can go undetected by either the failing component or any component communicating with it. NMESys periodically initiates these integrity checks to determine whether its current view of the state of the network is correct. If some error is detected, then the condition is handled just as a reported failure would have been.

3.0 THE METHODOLOGY OF NMESys

NMESys is currently implemented on a DOS machine. This PC is connected to the network so that it becomes the network monitor. The interface to the network is implemented in C. The alarm processor and user command processor are implemented in CLIPS. These are used for the interpretation and tracking of alarms. This design strategy makes NMESys more flexible since only the C component would need major changes if a new type of network were being monitored. NMESys has three major components: the alarm and user interface, the alarm processor, and the user command processor. The basic architecture of NMESys and the interaction between the components is shown in Figure 1.

3.1 THE ALARM AND USER INTERFACE

The alarm and user interface handles incoming alarms, timing for integrity checking, and the user input menu. This portion is written in C for three reasons. First, this task must be able to communicate with external devices. Second, it must be able to receive new alarms from the network as well as detect that the user has initiated a command from the menu. Third, it is the component in the system which implements time. Since NMESys is a prototype system and is not connected to a network at this time, the alarm and user interface also contains a random alarm generator. When an alarm is generated, it is passed to the CLIPS alarm processor. The current time is also sent to CLIPS so that degrading conditions beyond their threshhold can be properly determined. The user menu is also presented from this process. If a command is initiated, then the appropriate command request is passed to the CLIPS user command processor. In both cases, control is returned to the alarm and user interface when processing is completed. In addition, every five minutes a command is automatically generated for CLIPS to initiate any integrity checks which are due. This command tells CLIPS the current time so that the appropriate calculations can be made.

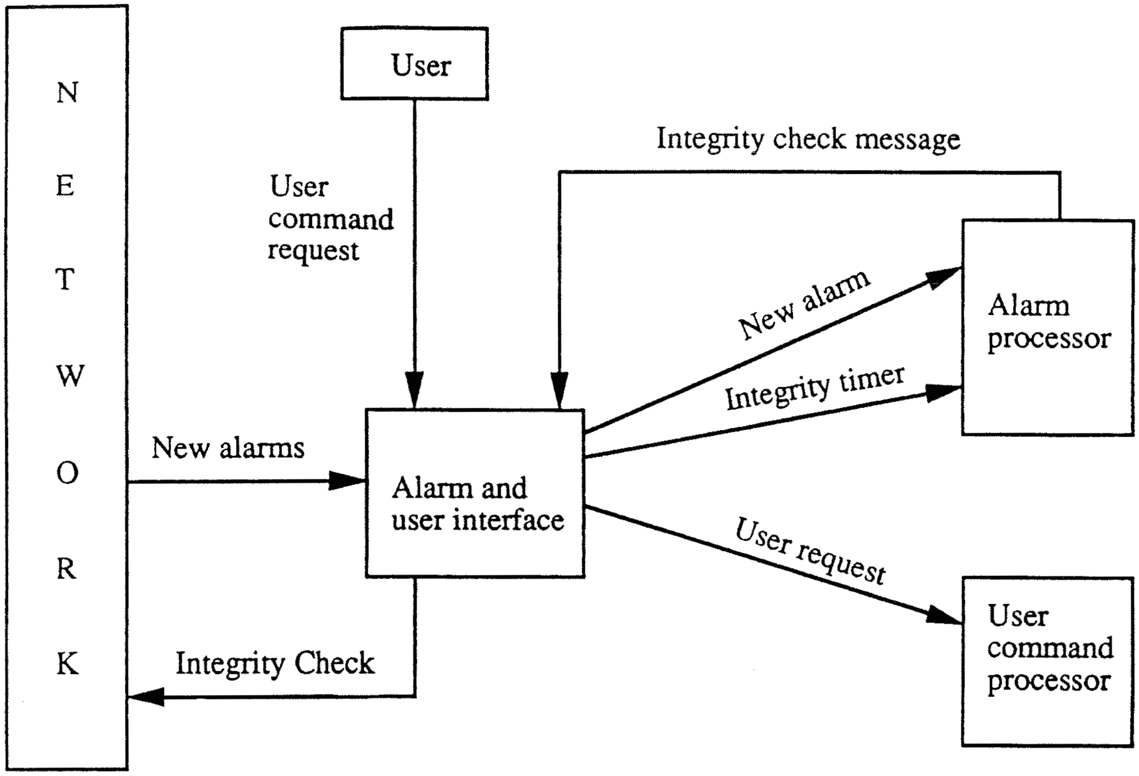

Figure 1. NMESys architecture

3.2 THE ALARM PROCESSOR

The alarm processor is written in CLIPS. It contains a list of all alarm types which can be generated by the network. Thus the new state of the component can be determined by the type of alarm which has been received. Since the alarm types and their characteristics are facts to CLIPS, implementation of a new type of network would involve only redefining the new alarms to CLIPS. The received alarm is always logged for site history purposes. If the alarm indicated a DOWN condition, an alert is generated and posted to the screen. The alert is logged so that an operator can always see the current list of alerts. If the alarm indicated a DEGRADING condition, then CLIPS checks if the threshhold for that alarm has been exceeded. If so, an alert is posted. If the alarm indicates an UP condition, any alerts which are active for this site are automatically removed from active status. Thus, if alerts have been logged for a condition, but service has been restored, the preceeding alerts disappear from the active alert list. If this were not the case, then an operator would be forced to investigate each condition, only to find that the outage has been restored. By watching the active alert list, the operators only need to look into conditions which reflect the current status of the network.

The alarm processor also handles integrity checking. Integrity checking provides an additional level of confidence in the accuracy of the status of the network. Without a method of detecting failures, the network management system is only as powerful as the failure detection process of its weakest component. When an integrity check message is received from the alarm and user interface, CLIPS determines the last time each site received an integrity check. If one is due, then the appropriate message is sent to the interface so that it can be in turn sent to the appropriate site. NMESys knows what type of equipment is at each site. It also knows the format of the proper integrity check message by the equipment type. These are the two pieces of information which must be sent to the interface. Once again, the implementation of a new piece of equipment or an entirely new network would only involve definition of the new type(s) of equipment and the appropriate integrity message.

3.3 THE USER COMMAND PROCESSOR

The user command processor is also implemented in CLIPS. The commands which are implemented in the user menu are shown in Figure 2. The command processor contains commands to show all sites which are UP, DOWN, or DEGRADED. The operator can also show all alarms for a particular site, show all active alerts, or acknowledge a particular alert. The commands to show UP, DOWN, or DEGRADED sites allow the operator to get a current status of the network at any point in time. This is also useful when a network user is reporting a problem. The operator can check the status to see if any of the received alarms could be causing the reported failure. There is also a command to view the history of alarms for a particular site. This is most useful when diagnosing a problem to try to determine the exact point of failure. Network operators typically must research failure conditions which have originated from either user complaints or from received alarms. When a network user calls in a complaint, the actual point of failure may not be as obvious as when an alarm is reported. Thus, a network operator must have the ability to view the history and status of many of the components in the network to determine the exact point of failure.

There are also two alert commands which are most useful to the network operator diagnosing problems from conditions which have been reported through the network rather than user reported problems. One of the principles of NMESys is that it can filter alarms which do not need attention from an operator. An alert is a detected condition which warrants action. Thus, an alert may be initiated by a single alarm, or by some combination of alarms which indicate a failure. The command to view all active alerts is probably the most utilized command. This allows the operator to see all events which require some action. The network operator's job is to resolve each of these events one by one. If no events are on the list, then all conditions have either been resolved or are in the process of being resolved. The command to acknowledge an alert allows the operator to tell NMESys that the alert has been recognized. This does not necessarily mean that the situation has been resolved. It might mean a repair technician has been dispatched or that the appropriate agency responsible for the equipment has been notified. By acknowledging the alert, the active alert list can be maintained as only the events which still require operator action.

```
            Menu

1) Show DOWN sites
2) Show DEGRADED sites
3) Show UP sites
4) Show all alarms for a site
5) Show active alerts
6) Acknowledge an alert

0) Exit

Enter selection:
```

Figure 2. The user menu

4.0 RESULTS

NMESys is a prototype expert system, programmed in CLIPS, used to perform the task of network management. CLIPS made the alarm processing and user command processing tasks quick and easy to write. Information parsing and error checking routines can be implemented in just a few lines of code. This made it very simple to write and test routines quickly. In the future, enhancements will be implemented and evaluated without a large investment in programming effort. This is obviously an excellent type of environment for prototyping systems. CLIPS can also interface with system level commands, in this case, DOS commands. Thus, CLIPS can be used in conjunction with any other tool or user program. If there is some task that CLIPS cannot accomplish, the task can be written in another language and interfaced to CLIPS. This was the case in NMESys where C was utilized to communicate with the network.

The use of DOS presented some problems, mainly because of the lack of multitasking. C was utilized so that NMESys could process network messages while waiting for user commands. The requirements of NMESys do not allow the use of the CLIPS read statement for user input. If the CLIPS read statement were used, then the operator could be at a read prompt while alarms were coming in from the network and NMESys would not be able to process the alarms. There would be no way to limit how long the operator remained at the prompt. All input from the operators had to be implemented in C and interfaced to CLIPS.

NMESys also brought out some application level discoveries and problems. It was initially thought that the status type commands would be most helpful to the network operator. However, it soon became apparent that the alert portion of the system was more helpful. The active alert list became the work queue for the network operator. This section of the system seems to have the most potential for expansion to further aid the network operator and produce more accurate diagnosis.

5.0 CONCLUSIONS AND FUTURE WORK

NMESys has a number of benefits to assist in the task of network management. First, the network operators's job is eased since NMESys does all the tracking of equipment states as well as events which require action. The system provides more accurate real time status than a human operator could provide. This allows the network operators to have timely information when network users call in to report problems. NMESys also maintains the work queue for the network operator by defining the active alerts. The network operators only need watch the alert list to determine what areas in the network need attention. NMESys is also flexible since a different type of network could easily be implemented by only changing the C interface and defining the facts for the types of alarms and equipment in the new network.

There are many potential enhancements which are planned for NMESys to increase its functionality. First, a chronic function should be implemented. This function would alert an operator when a site has posted too many alarms in a certain time period. For example, a site may post many alarms which clear very soon afterward. An operator might not even see the alarms if they clear quick enough since NMESys will remove active alerts when an UP condition is detected. Or, if different operators work on the problems, they may not realize that the site has actually exhibited many problems. Another condition which could occur is that the alarms are diagnosed as no trouble found situations. In this case, the operator might be able to clear the problem by executing some sequence of commands at the computer site. In both of these situations, the operators should be alerted that a particular problem is occurring over and over again. This would give the operators the indication that these are not random harmless failures, but possibly the result of a problem which is manifesting itself. Thus, similar to the threshhold concept for degrading conditions, sites which show many short spans of down time can be diagnosed as to the true cause of the failure.

Another enhancement which could be very helpful would be to expand how active alerts are purged from the list when an UP condition is detected. For example, an UP condition for some type of DOWN alarm might not always mean that all DOWN conditions have been restored. An example would be if a DOWN alarm indicated a failure on a particular connection to another computer and another DOWN alarm meant that the printer was not working. If the computer connection comes back UP, this does not mean that the printer has also been restored. NMESys should contain an alarm cross reference table to indicate which alarms are related. Thus, when an UP condition is received, only related DOWN alarms are removed from the active alert list.

Many enhancements could also be made to the functionality of the alert list. When an operator acknowledges an alert, a reason for the resolution should be requested from the operator. For example, the operator might indicate that the disk drive was replaced, or that there was no trouble found. This allows NMESys to begin to help the operator diagnose the potential cause of the failure before researching the problem. Over time, trends can be developed on the types of fixes which occurred depending on the reported alarm code. The operator could query NMESys on the past history of a certain alarm code. NMESys could indicate the percentage of each kind of resolution code for the history of alarms. This feature would be extremely helpful to the less experienced network operators. These are just a few of the areas where NMESys could further help the network operator to perform the job more accurately and efficiently. As NMESys continues to grow, more and more enhancements can be envisioned, each building toward a fully automated network monitoring and diagnosis system.

6.0 REFERENCES

[1] Callahan, P., "Expert Systems for AT&T Switched Network Maintenance", AT&T Technical Journal, Jan. 1988

[2] Giarratano, Joseph, Riley, Gary, "Expert Systems Principles and Programming", PWS-KENT Publishing Company, 1989

[3] Harmon, Paul, et al., "Expert Systems Tools & Applications", John Wiley & Sons, Inc., 1988

[4] Vesonder, G. et al., "ACE: An Expert System For Telephone Cable Maintenance", IJCAI (1983) pp. 116-121

[5] Waterman, Donald A., "A Guide to Expert Systems", Addison-Wesley Publishing Company, 1986