Proceedings of the 1988 Globecom Workshop on the Application of Emerging Technologies in Network Operation and Management, December 1988, Florida

NETREX
-
A Real Time Network Management Expert System

Rodney M. Goodman

John Miller
Padhraic Smyth

California Institute of Technology

Department of Electrical Engineering, Caltech 116-81,
Pasadena, CA 91125, (818) 356-3677

and

Hayes Latin

Pacific Bell

Information Systems Organization, Pacific Bell, Concord, CA 94520

IEEE, Globecom Workshop on the Application of Emerging Technologies in Network Operation and Management, Dec 1988, Florida, USA.

Abstract

NETREX is an expert system controller that works as part of an integrated network management system (ISM) being developed by Pacific Bell. NETREX interacts in real time with an environment of databases, and network manipulation and sensing systems. In this paper we outline the overall goals of the ISM and NETREX projects. In particular, we describe a prototype system which implements one aspect of NETREX, a trouble ticket expert. The prototype scans open (active) trouble tickets in real time and then proceeds to collect sense and status information from the network environment. A diagnosis is made, annotations are made to the ticket, and a repair is attempted if possible.

This work is supported by Pacific Bell.

1. Introduction

Large data networks are becoming unmanageable with human effort alone. Such networks are often widely geographically distributed and consist of many thousands of terminal devices. Switching centers can consist of dozens of nodes with numerous different vendor equipments and network protocols. The advent of LANs and the prospect of ISDN only makes the problem worse. In addition, even with the best intentions, network evolution is rarely planned but is implemented on an ad-hoc basis as user demand dictates. The task of fault isolation and repair in such an environment falls to the network management operation, and is very demanding in terms of human expertise. This expertise is limited by both lack of skilled personnel, and by the rapid pace of technological development. Repair specialists are faced with a rapidly changing multi-vendor network environment that produces huge volumes of automated device alarm messages. It is unreasonable to expect repair personnel to function effectively without a new and sophisticated set of tools.

In this paper we give an example of how real time autonomous expert systems can help to make these networks manageable. The key words here are real-time and autonomous. Most current knowledge-based expert systems operate in a "consultant" paradigm. That is, a user is supposed to sit down at a terminal and go through a "consultation session" with the expert system (ES). In a real trouble analysis center (and in many other real time decision making organizations) such a scenario is totally impractical. The ES users do not have the patience to answer the expert system's questioning as it tries to formulate a model of the problem - they want to be solving the problem. In addition the proposed users have a generally high level of skill, and will not tolerate seemingly abstruse questions. Thus the ES must work in parallel with the human system, and acquire its own picture of the problem environment. Having done so, the autonomous system can act in two possible ways. First, the ES can instigate its own problem repair completely autonomously, without consulting the human experts. Alternatively, the ES can interact with the human experts. If the latter course is taken then the ES must interact at a sufficiently "high level" if it is to gain acceptance by the human experts. The ES must therefore be integrated into the existing human operated management system. In order for such an autonomous system to be even possible in the network management environment, there has to be in-place an "effector" and "sensor" system with which the ES can observe and manipulate the network. It is vital that this system is in place and relatively stable, otherwise the ES design will devolve into this lower level design effort, as opposed to the higher level design effort needed for expert level problem solving. This situation mirrors that of early computerization: if a task that had an existing "manual" system it could be computerized efficiently and elegantly, whereas one that did not took much more effort and hence cost.

The use of expert systems in network operations and management has recently mushroomed [1,2]. ES technology is obviously appropriate to solving these problems, but as yet, most of the reported systems do not work in real time. However, this is changing rapidly as the economic benefit of real time expert systems becomes more widely appreciated.

NETREX (NEtwork Trouble and Repair EXpert) is our approach to automatically managing data networks with expert systems technology. NETREX is an on-going project with many facets, and intimately linked with the effector and sensor systems being put in place by Pacific Bell (P*B). The expert systems part of the project is a collaborative effort between Caltech and P*B. The mix of research and real world environments has been very successful in tackling this highly demanding domain of real time network management. In this paper we describe the overall goals of the project, and in particular the prototype system running at present. We also outline how we intend to develop NETREX further into a production system.

2.0 The Problem Domain

The problem domain we are dealing with is the management of P*B internal data networks. These networks support vital company applications such as billing, service orders, inventory and so forth. The efficient operation of these networks and the provision of very high uptime is of vital commercial value to the company. The hosts that support these applications are distributed in several major computer centers over California, each computer center contains hundreds of mainframe hosts. The networks that support these applications are varied (e.g. async, bysync, packet, LAN etc.) and large (e.g. 20,000 terminals on the BANCS bysync network).

The task of managing the networks and providing the system of trouble analysis and resolution takes place in several Trouble Analysis Centers (TACs). The software that supports this activity is ISM - Integrated System Management. The ISM software consists of three integrated parts NETREX, LINC, and TIMS. NETREX is the expert system manager which uses the two external programs LINC and TIMS to obtain information on the network environment. The LINC program provides the means to remotely manipulate and sense the network (the effector and sensor system). TIMS is a database that stores inventory, genning information, and supports the trouble ticket and alarms log infrastructure of the fault isolation and repair process. These two modules are very significant pieces of software in their own right and without them it would be *absolutely impossible* to implement the level of real time expert system we envision.

Consider the "manual" system as currently operated by the TACs. A client with a problem calls the center and explains the problem to a Data Specialist or "call screener". This is a task that involves interaction with the client and many vague problems are cleared up at this point. This task is absolutely not suited to an expert system because of the breadth of the problem domain. For example, anything from the power cord not plugged in, to coffee stains on the keyboard. If the problem is non-trivial a trouble ticket is entered into the TIMS data base. The Data Specialist may (depending on expertise) then try to resolve the problem by doing more tests. If the problem is resolved then a client premise "fixit" agency may be dispatched. If the trouble can not be resolved at this stage more sophisticated testing is necessary, and the Data Specialist passes the ticket on to the COMTECH section. In parallel with this a "statuser" monitors open trouble tickets and commit times to ensure proper prioritizing of the tickets and to deal with client interaction. Comtech personnel are very experienced in data communications and have a deep understanding of the topology of the network. Also, Comtechs have more sophisticated network access and data scoping equipment with which to probe the network. For example, the ANMACS system is a sensor and effector system accessible from LINC that enables data lines anywhere in the state to be broken for loopback and scoping tests. Also, automatic alarms are continually being logged by the TIMS/LINC system and displays of these are continuously available to the Comtech section. Using these tools the Comtechs can quickly dispense with problems of a "medium" level which may have been out of the capability of the data specialists. This leaves the really hard problems such as intermittent faults, or those that require significant client/fixit-agent interaction, and these may take considerable expert level diagnosis to solve. Finally, the problem may be resolved, in which case the trouble ticket is closed with the appropriate annotations; or the responsibility for the ticket is passed to another agency (e.g. plant control, or customer premise equipment).

It is clear that automation of the above procedures would result in significant benefits in terms of faster problem resolution and hence shorter client down time. Current problems that occur in the manual system are as follows. First, no preventative maintenance is done. The TAC is too busy fire-fighting in the day to do such work, and only a skeleton staff exists at night. Second, many problems are passed onto the Comtechs that are really low level problems. This occurs because of the vari-

able skill levels in the data specialist section. Also, many problems prove to be transient and simply "come clear". Thus valuable expert time is spent on relatively trivial problems. Third, the alarms displays provide too much information for the Comtechs to effectively use (the three mile island problem), thus valuable alarms information that could help problem resolution in difficult cases (e.g. intermittent faults) is often ignored. Furthermore, automated alarm driven diagnosis would enable problems to be resolved before they even resulted in a trouble ticket, and thus also save valuable Comtech time.

3.0 The NETREX system goals and scenarios.

The P*B ISM system is an approach aimed at solving the problems outlined in the last section. The ISM system is an integration of NETREX, TIMS, and LINC. In this section we outline what we envision the final form of NETREX to be. NETREX is envisioned as a collection of expert system modules sharing knowledge bases, databases, and network manipulation methods in order to provide a number of specific problem solution tools and methodologies. A fundamental assumption is that NETREX works and interacts with users and other software systems in real time. Furthermore, NETREX modules do *not* operate in a "consultant" paradigm in the traditional sense; but rather have a high degree of autonomy.

The types of NETREX modules can be split into two types: those that are completely autonomous; and those that interact with a human user. First, consider those modules that interact with a human user.

Data specialist's assistant: One of NETREX's modules works in the background of the data specialist's terminal. As soon as a trouble ticket is opened NETREX starts its own data gathering and analysis of the problem. NETREX "shadows" the data specialist and offers relevant data and its own problem analysis , if appropriate. For example, if there were any automatic alarms generated on this device or its peers in the hierarchy. Or previous trouble reports and their resolution. For example, NETREX can quickly say "the client's screen is dead because I have an alarm that 20 minutes ago the processor hosting the application went down".

Comtech workstation: this is a development of the above assistant, that would aim to integrate all the scoping and network management tools into a fully windowed environment. The module would interact with the Comtech at a high level, and be able to answer complex hypotheses. For example, "what will be the effect on site X if I take node 3 out of service now?". Such impact analysis could have applications in load planning and network analysis.

Field Repair Assistant: This module would interact with field repair agencies as they try to fix problems at customer premises. The module would allow network manipulation and test without tying up the skilled comtechs.

The second type of module is the type that operates without human intervention. The objective here is to automate the network problem resolution and repair process as far as possible.

Alarms monitoring and repair expert: This is perhaps the most important and real time intensive operation to be attempted. Alarms from all network components are being continuously monitored and stored by the LINC process. Alarms would be filtered and passed to the alarms expert in real time for processing. Raw alarms would be processed into significant "events" some of which would be simple, such as indications of processor overload, to a sophisticated analysis of a recurring and intermittent trouble. Problem resolution would also be undertaken by this module. Repair scripts would be automatically evoked to clear the trouble. These would range in complexity from simple restarts of network components, to complex sequences needed to bring

up failed network hosts. Becoming alarms driven brings both benefits and problems. The expert system must now be truly real time in a "machine time" sense, as opposed to a "human real time" system.

Trouble ticket expert: This module would be positioned between the data specialists and the Comtechs. The objective would be to remove trouble tickets from the stack being queued for Comtech attention. If the problem can be cleared by the ES the ticket is removed from the stack, again saving valuable Comtech time. The module would also perform statuser and escalator checking of pending trouble tickets, making sure tickets are not "forgotten". This is particularly important when responsibility for the ticket is shared among various agencies.

Network Status and Configuration Monitor: This module has responsibility for maintaining an accurate model of the network in the face of changes due to genning new equipments into the network, as well as outages. The module would do automatic health monitoring during the night. A desirable characteristic of this module would be the ability to automatically learn about the operation and configuration of the network. This information would be of use in impact analysis.

4.0 The NETREX prototype - design criteria and goals.

We have implemented and tested a prototype module of NETREX. This module is the trouble ticket expert outlined in the last section. The objective of implementing the prototype was to test the feasibility of the NETREX plan, and to demonstrate various "proofs of principle". In particular to develop the real time aspects of the system such as the querying of remote databases. In this section we describe the various design criteria that were adopted in order to achieve a working prototype in one year.

The first major constraint was that the system would be developed at Caltech in Pasadena, and would thus have to remotely access TAC facilities in San Diego. Even though the final system will be integrated into TAC hosts, we felt it necessary to demonstrate that the "brain" of such a system could be remote from its sensors and manipulators into the network cloud. This constraint allowed us flexibility in determining the implementation platform. We chose to implement the system on a Unix SUN 3/160 workstation. This decision allowed us access to sophisticated programming tools in C, and easy upward migration for more compute power. Much of the programming effort in a real time system is in the interfaces, and C and Unix provide a powerful environment for programming these interfaces.

The next major choice was the choice of an expert system shell. We chose the Teknowledge S1 shell, for a variety of reasons. First, it is written in C and has interfaces into C and Unix. These are essential for the development of a real time system. Second, S1 was available for the Tandem Computers used in the TAC, and we wanted to keep open the possibility of porting to these machines in the future. Third, the S1 syntax has a very readable Pascal style. This is important for transferring the technology over to P*B programmers. Although S1 did not have all our requirements for a suitable shell (non-monotonicity is a problem), these other factors far outweighed the limitations. Also, because of the easy interface to C many of these limitations could be bypassed by C programming. Finally, a new upward compatible improved version of the product called Copernicus would be released by the time a production prototype would need to be fielded.

A major choice of operational paradigm was that the ES would emulate an expert human user in the sense that standard interfaces to P*B applications would be used. Thus the ES would interact with the applications via terminal emulation. This relieved P*B from having to provide any special interfaces into their applications. This was a major benefit in development time, although the resulting interfaces were not as fast as they would have been app to app.

Having decided on the above constraints, we were in a position to define the operation of the trouble ticket expert module as follows. The ES would monitor the CCTAC database and look for open (active) trouble tickets. The ES would then try to diagnose the problem. The final result would initially be an annotation of the trouble ticket with the ES's findings, leading on to pro-active repair of the fault (if possible). In this way the findings and performance of the ES could first be assessed by comparison with closed out tickets. In order to achieve this performance the ES would need to interact with the trouble ticket database, the alarms database, and to issue active commands to probe the network.

We then narrowed the domain of the problem to deal with only one network - the BANCS bysync network. There were several reasons for this decision. First, the BANCS network is one of the largest within P*B and is well established with many CCTAC experts available. Second, the new LINC and TIMS applications were still undergoing modification and testing. Thus although they could be accessed via the BANCS network we would not have to rely on them being stable initially. The final prototype that we have implemented is thus a BANCS expert module which can interrogate the NDBS database for inventory and trouble ticket information, can access alarms information, and can issue active THP commands to obtain current sense and status information. Figure 1 shows a block diagram of how the system interacts with the network.

5.0 The NETREX prototype - architecture and operation.

The ES acts as a regular terminal user. The facilities at the Caltech (development) end therefore consisted of an IBM3274 control unit with a standard 3272 terminal, as would be used by a CCTAC data specialist. Also coming off the CU is a printer, and the connection to the SUN. The SUN runs a commercially available 3272 emulation program that handles the low level terminal protocols and enables several virtual terminal sessions to be run simultaneously. The structure of the SUN software is shown in Figure 1. At the highest level the SUN is running UNIX, and the S1 shell which runs the prototype ES runs within that. When the ES needs external information a call is made to a data cache program written in C and running as an autonomous process within Unix. The data cache program handles both the dialogue with S1, and the terminal emulator. In addition, information is buffered in the cache so that unnecessary requests to P*B databases are not made. This is essential given the low speed link to P*B facilities in this prototype. However, any ES needs some form of externally programmed dialogue manager and local database if it is to operate in real time. It is far too inefficient to attempt this process within the shell itself.

The cache system is composed of a database program, a set of remote-command functions, and a set of filter programs written for the UNIX *awk* tool. The database stores an arbitrary number of datasets. Each dataset can have any number of hash tables. These tables index the dataset by different keys. The remote-command functions are executed by the database program whenever a query would result in a 'not found' being returned. Instead of returning not found, the command-function formats a command or sequence of commands to the remote mainframe. The results of this query to the mainframe are passed through the awk filter and loaded into the database. The original query is attempted again. This time the results are returned directly even if the query result is 'not found'.

The cache system serves two purposes. The first one is to reduce the number of queries to the remote system. The queries from the expert system to the cache system are for one field in a large page of data. Since the expert system is likely to request other fields from the same page, the cache system usually does not have to execute a remote command to get the required data. The second purpose of the cache system is to insulate the expert system from changes in the interface at the remote end. The expert system does not have to be concerned with how the information is obtained. Changes in the screen format of output data from the remote mainframe

require a change in the small awk filter program for that dataset. Changes in the syntax for queries to the remote mainframe require a change in the appropriate remote-command function. The entire cache system can be extended by adding the required remote-command function, writing an awk filter if it is needed, and changing the database table which describes the names and locations of database sets and database fields.

6.0 The NETREX prototype S1 program

Much of the initial work in developing the NETREX prototype was in the interfaces, as opposed to intensive knowledge engineering. The rule based knowledge came from both Comtech experts and equipment manuals. We felt that it was most important to get the structure of the expert system and its interfaces correct first, and then to develop the knowledge base later.

NETREX has been implemented using the S1/Copernicus expert system shell from Teknowledge. The knowledge representation scheme employed in this shell is based on object-attribute-value triples. Essentially, objects are specific items which the expert reasons about when solving problems, attributes are characteristics of these objects, and values are possible values of the attributes. The basic paradigm of operation is the determination of attribute values for specific instances of objects, via backward rule-chaining. Objects may range from specific physical devices (e.g., a printer or a multiplexer) to abstract notions such as "problem" or "intermediate symptoms." Figure 2 shows some of the S! NETREX rules.

In the NETREX trouble ticket prototype we perform diagnosis of "network problems". NETREX monitors the local cache database for new "open" (unresolved) trouble-tickets. The local cache database significantly simplifies the problem of interfacing the expert system shell to the real "live" network, effectively buffering the two. Once a new ticket is detected, NETREX creates an instance of the ticket and then determines many of the attributes of this ticket in a procedural manner. For example, network ID data is extracted and a search is made for the most recent trouble ticket relating to this device. The ES also parses various parts of the free form notes section of the ticket to try to assess what the status of the human problem resolution on this ticket. For example, has it been referred to another agency? It is feature of this shell that procedural code can be incorporated into the general inference scheme - in this case we found that the experts invariably "eyeball" some initial information about the problem before considering any hypotheses.

The device on which the ticket is opened is then checked via a series of THP status commands both to verify the information on the ticket and to determine the status of surrounding and hierarchically-connected components. As an example the "DSS" (display station status) command returns a list of abnormal conditions (line error codes, poll fail messages) both for the suspected device and related devices. The time of failure (as reported on the ticket) is also used to search the alarm log for any prior alarms for the device - this simple procedure is easy to perform in an automated system but is rarely used by the experts because of interface difficulties. At this point NETREX begins to diagnose the problem by backward chaining on three different attributes of the instance "problem", namely, the level, nature and current status of the problem. Additional network commands may be invoked during this backward chaining if more data is required. Based on these intermediate attributes NETREX finally backward chains to determine the attibutes "actual.problem" and "recommended solution". The result of the problem resolution is logged on a local copy of the ticket for future analysis. Figure 3 shows an examples of the output of the NETREX prototype program.

7.0 Conclusions

The NETREX BANCS prototype has satisfactorily demonstrated several important proofs of principle, validating our design methodologies. The most important of which was the real time accessing of external databases and the issuing of active network commands. The "expertise" shown by the prototype was secondary. However, it has emerged that data collection, digestion, and presentation are well suited to ES technology. Also, the prototype was capable of useful if low level diagnosis and repair (say recycling a line to bring it up again). The prototype at the time of writing is still being assessed, and its knowledge base significantly increased. Initial performance of this simple version of NETREX has been quite encouraging and with the addition of more sophisticated techniques such as the correlation of multiple problems (due to one source) it is expected to deployed in a production version within the next twelve months. NETREX prototype 2 is concurrently under development and will incorporate even more powerful features. Firstly, the C interfaces to the TIMS database and LINC control system are being developed, and these will function app to app via X.25 links. This will open up high speed access to more networks and components and enable a more sophisticated network model to be built up by the ES. Second, the system will become alarms driven. Also, prototype 2 is being written in Copernicus - the updated shell. This will allow more efficient dialogue with the C and Unix environment, and provide much needed ES features such as non-monotonicity. Overall, given our experience in developing the prototype, we feel that the design goals for the full NETREX system outlined earlier are quite feasible with current ES technology.

8.0 References

[1] "Knowledge Based Systems for Communications," IEEE Journal on Selected Areas in Communications, Vol. 6, No. 5, (ISSN 0733-8716), June 1988.

[2] "Expert Systems in Network Management," IEEE Network Magazine, Vol. 2, No. 5, Sept. 1988.

BANCS PROTOTYPE

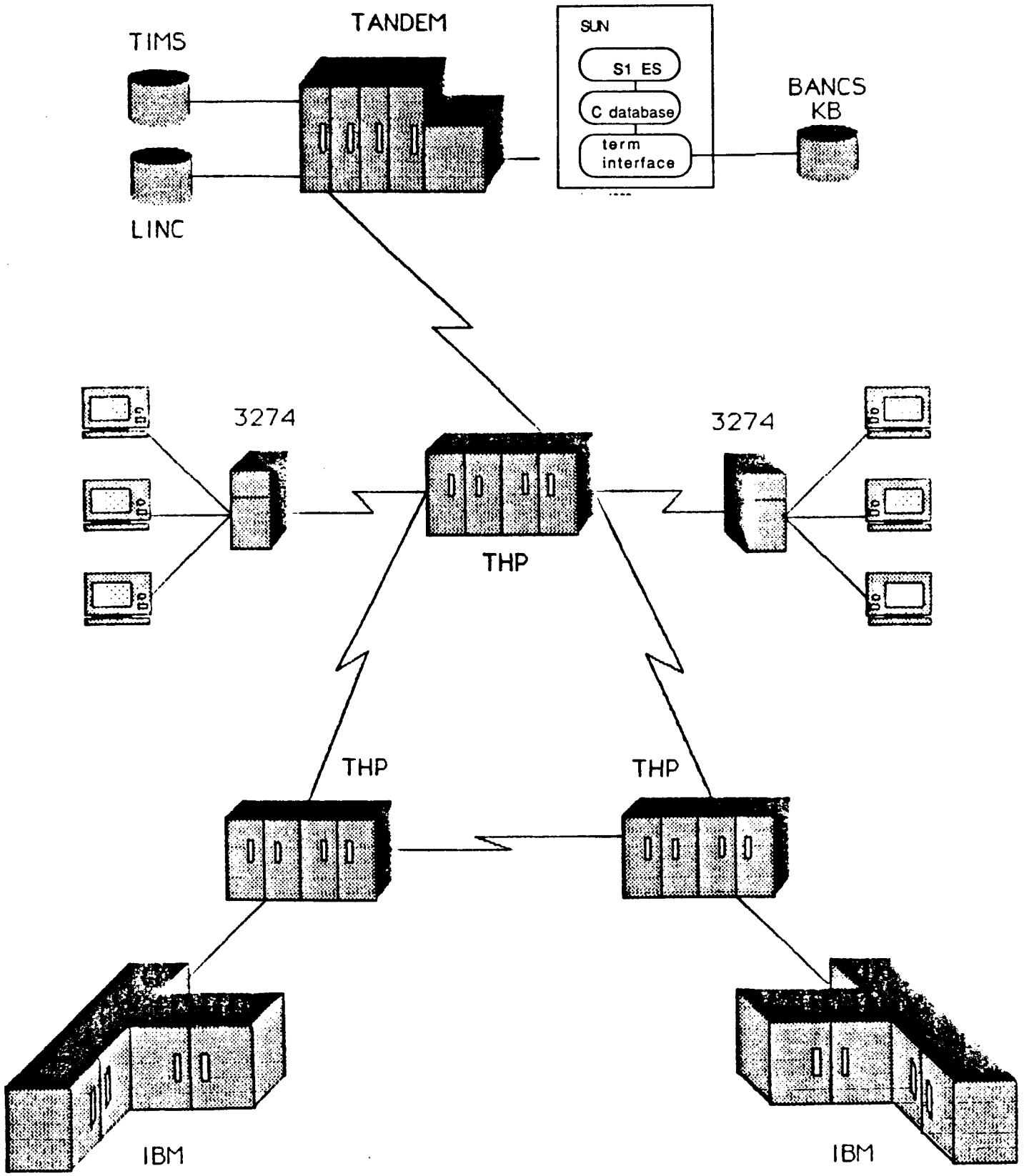


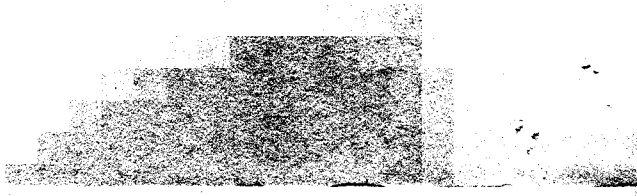
FIGURE 1

```
DEFINE RULE             DSS002.1
::APPLIED.TO            p:prob
::PREMISE               already.existing(t:terminal | last.terminal(t) and
                        cfl.count[t] > 3)
::CONCLUSION            nature[p] = "related to other devices on the same
                        control unit" <0.6>
END.DEFINE
/* Internal Slots for dss002
::CONCLUDED.ATTRIBUTES {nature}
*/

DEFINE RULE             DSS002.2
::APPLIED.TO            p:prob
::PREMISE               already.existing(t:terminal | last.terminal(t) and
                        cfl.count[t] > 3)
::CONCLUSION            level[p] = "line" <0.5>, "control.unit" <.6>
END.DEFINE
/* Internal Slots for dss002
::CONCLUDED.ATTRIBUTES {nature}
*/


DEFINE RULE             DSS003
::APPLIED.TO            p:prob
::PREMISE               already.existing(t:terminal | last.terminal(t) and
                        dss.modifier[t] ~= "no polling problems")
::CONCLUSION            current.status[p] = "there is a polling problem"
END.DEFINE
/* Internal Slots for dss003
::CONCLUDED.ATTRIBUTES {current.status}
*/

DEFINE RULE             EBC4050.1
::APPLIED.TO            p:prob
::PREMISE               already.existing(t:terminal | last.terminal(t) and
                        match.strings(alarm1.string[t], "EBCDIC STATUS
                        *4050*"))
::CONCLUSION            current.status[p] = "the device does not exist from the
                        CU's viewpoint"
END.DEFINE
/* Internal Slots for ebc4050.1
::CONCLUDED.ATTRIBUTES {current.status}
*/

DEFINE RULE             EBC4050.2
::APPLIED.TO            p:prob
::PREMISE               already.existing(t:terminal | last.terminal(t) and
                        match.strings(alarm1.string[t], "EBCDIC STATUS
                        *4050*"))
::CONCLUSION            level[p] = "station"
END.DEFINE
/* Internal Slots for ebc4050.2
::CONCLUDED.ATTRIBUTES {level}
*/
```

FIGURE 2

```
We are looking at a new trouble
This trouble ticket will be referred to as ticket 6
 The time currently is 1412

Ticket data obtained from NDBS :
 ticket 6  field:trouble ticket number value:304314
 ticket 6  field:device                value:F026207
 ticket 6  field:component type        value:STAT
 ticket 6  field:device type           value:4540PT
 ticket 6  field:date originated       value:08 16 88
 ticket 6  field:time originated       value:07 54
 ticket 6  field:ticket refered to     value:OCS
 ticket 6  field:previous ticket no.   value:
 This ticket has been open for 1 days and 6 hours and 18 minutes
 The most recent ticket for this device is not stored in the NDBS database

 THP report data on terminal #4 :
 terminal #4  field:network address value:F026207
 terminal #4  field:terminal type   value:4TP
 terminal #4  field:application     value:D58
 terminal #4  field:log.status      value:ON
 terminal #4  field:inactive        value:
 terminal #4  field:autolog         value: X
 terminal #4  field:dss result      value:no polling problems

 Alarms data from NDBS:
 terminal #4  field:alarm message    value:CAL FAIL NAK
 terminal #4  field:alarm message    value:EBCDIC STATUS *4050*

 Analysis of the data :
The problem appears to be at the station <.9> level
The nature of the problem appears to be: transmission errors <1.0>, and severe
<.3>
The current status is that there are no polling problems with the device
<1.0>, the device is logged on <1.0>, the device is CFL due to negative
acknowledgments to a THP message <1.0>, and degraded service <.6>
```

FIGURE 3



ITRULE

ITRULE is being used with experts to:

*** Identify topological rules:**

IF control_unit a is down THEN terminals xaa down

***Learn repair scripts:**

IF terminal_status = fail AND control_unit NOT down THEN disable/enable line

*** Identify temporal loadings:**

IF time=(>9.30am AND <10.30pm) AND NODE1=alarm THEN NODE2=alarm in 15minutes with prob=.8

***Analyze Trouble tickets :**

NODE	STN	Dev_typ	SYMPTOM	DEV_S	DEV_no	AL_ST	RC_CU	IL_XIL	DL_XD	BCT	Ref1	Ref2	Close_Cat	Soln_Dev	Soln_Act	Soln_By	Fix_By	time_down
D0	40	4540A	?	CUF	ALLDEV	*	OK	*	*	*	*	*	TERM	CU	RC	CS	CLIENT	under1hr
D0	C1	TC174	?	CFL	ONE DEV	*	OK	*	*	*	*	*	TERM	CU	RC	CS	CS	under1hr
B0	C4	4540A	NO CO	*	*	OK	NOK	*	*	*	PCO	*	NO TRBL	*	*	*	*	over3hrs
B0	C1	TC174	NO CO	CUF	?	*	*	*	*	*	PCO	*	NO TRBL	*	*	*	*	over3hrs
B0	40	4540B	NO CO	*	*	*	*	*	*	*	PCO	*	NO TRBL	*	*	*	*	under3hrs
B0	40	TC174	NDE DWN	CUF	ALLDEV	*	*	*	*	*	TESTER	*	TELCO	LINE	?	?	?	under1hr
B0	40	4540A	?	CUF	ALLDEV	*	NOK	NOK	*	*	TESTER	OCS	TELCO	LINE	?	TESTER	PCO	over3hrs
B0	40	SCC	?	CUF	ALLDEV	*	*	*	*	*	TESTER	*	NO TRBL	?	*	*	*	under3hrs
B0	C1	4540A	?	CUF	ALLDEV	*	*	*	*	*	OCS	*	TERM	CU	HRDWR	OCS	OCS	over3hrs
B0	C3	4540A	S/R	?	?	*	NOK	*	*	NOK	TESTER	*	TCU_SWR	CU	SFTWR	TESTER	TESTER	under1hr
B0	C1	TC174	INTRMCU	?	?	*	*	*	*	*	OCS	*	TERM	D/S	LOOPBK	OCS	OCS	under3hrs
B0	40	TC174	?	CUF	ONE DEV	*	*	*	*	*	OCS	*	DATA_SET	D/S	HRDWR	OCS	OCS	over3hrs
B0	40	TC174	INTRMCU	*	*	*	*	*	*	*	OCS	*	TERM	CU	HRDWR	OCS	OCS	over3hrs
B0	C1	4540A	?	CUF	ALLDEV	NOK	*	NOK	*	*	TESTER	*	DATA_SET	D/S	CONFIG	TESTER	CLIENT	under1hr
F0	C1	4540A	?	CUF	ALLDEV	*	NOK	NOK	*	*	PCO	*	TELCO	TCXR	HRDWR	PCO	PCO	over3hrs
D0	40	4540A	ALL_DWN	CFL	SOME_DEV	*	NOK	*	OK	*	TESTER	*	TCU_SWR	LINE	SFTWR	TESTER	TESTER	under1hr
D0	40	3274	TRM_DWN	*	*	*	*	*	*	*	OCS	*	TERM	CU	HRDWR	TESTER	OCS	under1hr
F0	40	554612	?	LMF	*	*	*	*	*	*	TESTER	*	TCU_SWR	LINE_MOD	SFTWR	TESTER	TESTER	under1hr
F0	40	4540A	?	CUF	ALLDEV	*	*	*	*	*	TESTER	*	TERM	CU	HRDWR	OCS	OCS	under1hr
F0	40	4540A	ALL_DWN	*	*	*	NOK	OK	OK	*	TESTER	*	TCU_SWR	LINE	SFTWR	TESTER	TESTER	under1hr
F0	40	4540A	ALL_DWN	CUF	ALLDEV	*	NOK	NOK	*	*	TESTER	*	NO TRBL	LINE	*	*	*	under1hr
F0	40	4540A	S/R	CUF	ONE_DEV	*	OK	*	*	NOK	TESTER	*	MUX	MUX	RESET	TESTER	NOG	under1hr

RULES :

	IF	THEN	p			J	
1	Solved_By OCS	Fixed_By OCS	1.000	0.308	0.346	1.53051	0.47093
2	Close_Cat NO_TRBL	Fixed_By *	1.000	0.192	0.192	2.37851	0.45741
3	Soln_Actn SFTWR	Fixed_By TESTER	1.000	0.154	0.154	2.70044	0.41545
4	Close_Cat TCU_SWR	Fixed_By TESTER	1.000	0.154	0.154	2.70044	0.41545
5	Fixed_By OCS	Solved_By OCS	0.889	0.346	0.308	1.06719	0.36941
6	Fixed_By PCO	Close_Cat TELCO	1.000	0.115	0.154	2.70044	0.31159
7	Soln_Actn RC	Solved_By CS	1.000	0.077	0.077	3.70044	0.28465
8	Fixed_By OCS	Soln_Actn HRDWR	0.889	0.346	0.385	0.79991	0.27689

FIGURE. 4