Network Expert Diagnostic System for Real-time Control

Terry L. Janssen

CSC Technology Center
Computer Sciences Corporation
3160 Fairview Park Drive
Falls Church, VA 22046

ABSTRACT

Data communications networks are controlled by network management systems that are responsible for performance and fault management. This paper presents an expert system capable of performing fault and performance management through different levels of autonomous control. A blackboard architecture design provides for processing of multiple lines of machine reasoning and planning: the set of all unresolved events is used to generate hypotheses of network state through event correlation and ancillary network information; supporting network data provides evidence that supports or refutes the hypotheses; conclusions are drawn from the hypotheses; plans of corrective action are built, executed, and monitored to attempt improvement to a "normal" network state.

1. INTRODUCTION

Computer networks are controlled by network management systems that are responsible for performance and fault management. They perform these functions by monitoring events and giving commands to various network devices such as terminals and host computers (along with their embedded communications capabilities) located on the networks. The "current state" of the network, on both the global and device levels, is monitored through event messages sent from the devices. Events range from device specific-faults to excessive protocol errors caused by multiple device interactions.

As data communications networks have become larger and more complex, the knowledge required to maintain them at acceptable levels of performance has likewise increased in complexity. This problem is compounded by networks of varying architectures linked through gateways and multiple vendors, network hardware (Boyd, *et al* 1987). Thus, network management is both complex and expensive, and network problems are often misdiagnosed, leading to excessive downtime.

Artificial intelligence has been applied to network management in a variety of areas. Generally, these efforts fall into one of the following categories: expert advisors that assist a network operator in performing network fault management, and expert managers that directly perform the network fault management functions. Cronk, *et al* (1988), provides a recent summary of some expert systems in the first category.

The success of expert systems in the first category may be accounted for by the success of expert diagnosis systems in related problem domains. Several expert systems provide fault diagnosis of electronics circuits (Bandler, *et al* 1985; Cantone, *et al* 1984; Merry, 1983; Laffey, *et al* 1984; Fredman, 1985), and central processing units (Gikes, *et al* 1986). ACE, a very successful expert system for diagnosing failures in telephone cables, correlates network fault indicators with customer trouble reports by off-line batch processing of historical data (Bernstein, *et al* 1988). Ganesan, *et al* (1988) developed an expert system on the front end of a network management system to allow the network operator to interact with a network management system using natural language, text and graphics.

The second category, expert manager systems, has been widely discussed as the next major evolutionary step in network mangement (Bernstein, *et al* 1988; Ward, *et al* 1985).

The ultimate goal of an expert manager in a data communications network is self-healing of failures and performance anomalies; when self-healing is not possible, an expert manager should automatically create a trouble report. Sutter, *et al* (1988), has suggested an approach to designing expert systems for real-time self-correcting networks.

Expert manager systems have not had the same degree of success as have expert advisor systems. One contributing factor has been that network management systems have not yet provided the capability for an expert system to access a

wide range of network data and statistics in multiple vendor and protocol environments. Also, network management systems have not provided for direct expert system issuance of tests and commands on-line to the network, and for operator control of which tests and commands the expert system should perform autonomously. Most network management systems in operation today do not have a network database including device information, network configuration and statistics. Such data is required for an expert manager system; without this information it appears that event correlation alone provides the greatest promise for on-line automated fault diagnosis in large data communications networks. Correlation of events within a data communications network has been suggested as a means of eliminating unnecessary diagnostic effort (Boyd, 1988).

Our research focuses on the development of an operator-independent, intelligent event processing capability for diagnosis and control. This capability is accomplished by integrating the development of a machine reasoning component into a network management system. Integration has been accomplished by consolidating this effort with the Integrated Network Management Control (INMC) system, another CSC Independent Research and Development effort. INMC provides a unified network management protocol and the network management functions for (1) access to events and statistics from the network, (2) performance of on-line network tests, and (3) issuance of commands to control devices on the network. INMC is an implementation of a network manager that uses the Common Management Information Service and Common Management Information Protocol developed by the International Standards Organization (ISO #9595/2, ISO #9596/2). The network management functions are being implemented in the C language on a MicroVAX. Networks from multiple vendors conforming to IEEE 802.3, OSI, and other protocols, either standalone or connected by gateways, are targeted for management by the INMC.

Our research and development effort has led to a total design for an expert system and INMC as an integrated system. The expert system, *i.e.*, the Network Expert Diagnostic System for Real-time Control (NEDS/RC), is targeted at performing most of the fault management functions of a human operator: near real-time event correlation and processing, diagnosis, and on-line network control for restoring a network state to "normal."

2. EVENT PROCESSING FOR FAULT DIAGNOSIS AND CONTROL

This section defines the fault diagnosis and control problem in terms of sets of object and their inter-relationships. An event is an indication of a fault. An event is either active, meaning that it is being processed, or inactive, meaning that it has been processed and is no longer an indicator of the current network state.

Let E be the set of active events:

$E = \{e_1, e_2, ..., e_i\}$ where i is the number of events, $i \geq 1$.

As events arrive they are added to the set; when resolved they are removed. For every event there exists at least one hypothesis h such that h is a plausible explanation of that event. Let H be the set of hypotheses:

$H = \{h_1, h_2, ..., h_j\}$ where j is the number of hypotheses, $j \geq 1$.

A hypothesis states that a specific failure or performance anomaly has occurred. Hypotheses are generated by event correlation and inference from ancillary network information. When possible a hypothesis states the specific point in the network where the failure or performance anomaly has occurred. The set of hypotheses suggest a set of plausible, unique states that may or may not represent the state of the network. Hypotheses are not redundant: only one hypothesis exists for a given plausible network state, and one or more events in set E is an indication of a specific hypothesis in set H.

Hypotheses are supported or refuted by evidence gathered by hypothesis testing. Tests can provide statistical or physical evidence. A database of network data provides traffic and error measurements for each level of protocol and of each network and subnetwork. Statistical tests are performed by utilizing network data. Physical tests are performed by sending messages to one or more devices on the network and executing a physical device test. Supporting evidence is sought for each hypothesis in set H by searching for a test t that can provide that evidence. Let T be the set of tests:

$T = \{t_1, t_2, ..., t_k\}$ where k is the number of tests, $k \geq 0$.

Each test in T provides support for one or more hypothesis in H, as measured by degree of confidence. Degree of confidence is a measurement of the likelihood that the conclusion is true. If the degree of confidence surpasses a threshold level of confidence (set by a parameter), the hypothesis that it supports is considered true. For every hypothesis in H with supporting evidence greater than the threshold of confidence, a conclusion is created and added to the set of conclusions C:

$C = \{c_1, c_2, ..., c_m\}$ where m is the number of conclusions, $m \geq 1$.

A conclusion is a belief that a particular fault has occurred. Each event in E (the driving force of this event processing system) has at least one conclusion in C. The likelihood that a conclusion c accurately models network state is measured by the degree of confidence in the hypothesis from which it originated. When possible, a conclusion points to a specific point in one of the networks or subnetworks where the failure or performance anomaly is believed to have occurred. For any event e, if no hypothesis h exists with a degree of confidence above the acceptable threshold of confidence, a default conclusion

is formed: the conclusion that the cause of the event is unknown (and the creation of a trouble report is required).

Conclusions need to be acted upon to attempt to restore the state of the network, *i.e.*, the state of the network that is believed to exist, to a "normal" network state. Normal is defined (in the context of this paper) to mean that the state of the network is consistent with network design and intended network performance.

Planning is required to restore network state to normal. A plan of action for the entire network is actually a set of subplans refined into a unique set of on-line commands for controlling devices within the global network. Let S be the set of subplans:

$S = \{s_1, s_2, ..., s_n\}$ where n is the number of subplans, $n \geq 1$.

A subplan is a part of the overall plan of action, and provides a means to change network state at a specific point in the network. For every conclusion in C there exists at least one subplan in S that can potentially improve network state to a more "normal" state. A subplan s may restore more than one network failure or anomaly concluded in set C.

Resolution management involves adding new subplans to the set S and refining the suplans into a total network plan composed of a string of nonredundant device commands. At periodic intervals the plan is executed by sending it to the host network management system for transmission on the network. The plan includes commands for feedback to monitor for network state improvement. The management of unique sets of events, hypotheses, tests, conclusions, and subplans provides for synchronization of multiple activities for managing multiple faults and performance anomalies occurring simultaneously within the network.

3. LEVELS OF AUTONOMOUS CONTROL

One major obstacle in development of an expert management system is maintaining control of the expert system, *i.e.*, avoiding inappropriate commands from being issued to one or more of the networks by the expert system. Expert system advisors leave decision making and command of network functions to the discretion of the human operator. Conversely, expert managers perform commands automonously. Events provide uncertain information, and the knowledge in a knowledge-based system is often inexact. Consequently, an expert management system is prone to making wrong inferences and executing commands that are inappropriate.

Three basic situations in expert management systems mandate placing control restrictions on the expert system: the expert manager can perform actions on the network (tests and commands) that are inappropriate; the expert manager can perform actions on the network that are appropriate to improving network state but require inappropriate amounts of network resource; tests or commands issued by the expert system may take an excessive amount of time to perform. The first occurs because of an entry in the knowledge source that is incorrect or incomplete. The last two occur when the knowledge is correct but not reasonable given current conditions.

Network resource utilization can be measured in terms of network or device load. Both network utilization and time are included with knowledge of specific tests and commands within the expert managers, knowledge sources.

One solution to the problem is to give an expert manager different degrees of autonomous control. Such an approach allows the operator to set the degree of autonomy that the expert system can assume. The level of autonomy can be increased over time as trust in the expert manager increases. Knowledge of network utilization requirements and estimates of time requirements to perform tests and commands provide rule-based constraints on the tests and commands that can be performed. Before a test or command is issued to the network it must surpass a threshold level that permits it to be executed without network operator approval.

This approach provides a continuum of control: control ranges from total control by a human operator through human operator override of the expert system's recommendations to complete autonomy of the expert system.

4. APPROACH

We reviewed commercially available expert system shells at the beginning of our research effort. None of the evaluated shells provided a good match for near-real-time processing; the correlation and processing of multiple events; and multiple threads of control needed to resolve multiple network failures and performance anomalies. All are required for autonomous network management.

Our approach has been to design an expert system to perform event processing for fault diagnosis and control, as defined previously (in Section 3). Design of NEDS/RC has resulted in a new expert system architecture that has evolved from other blackboard expert systems (Hayes-Roth, 1985; Craig, 1986). The blackboard is a repository of information within a global memory area. Distinct functions are performed on objects on the blackboard by independent knowledge-based processes. The processes manipulate five sets of objects on the blackboard: (1) objects that correspond to events that arrive from the network through a network management system; (2) hypotheses about fault and anomalous states within the network; (3) tests to support or refute the hypotheses; (4) conclusions formed from hypothesis testing; and (5) subplans to restore network state.

This approach provides several advantages. First, each black-

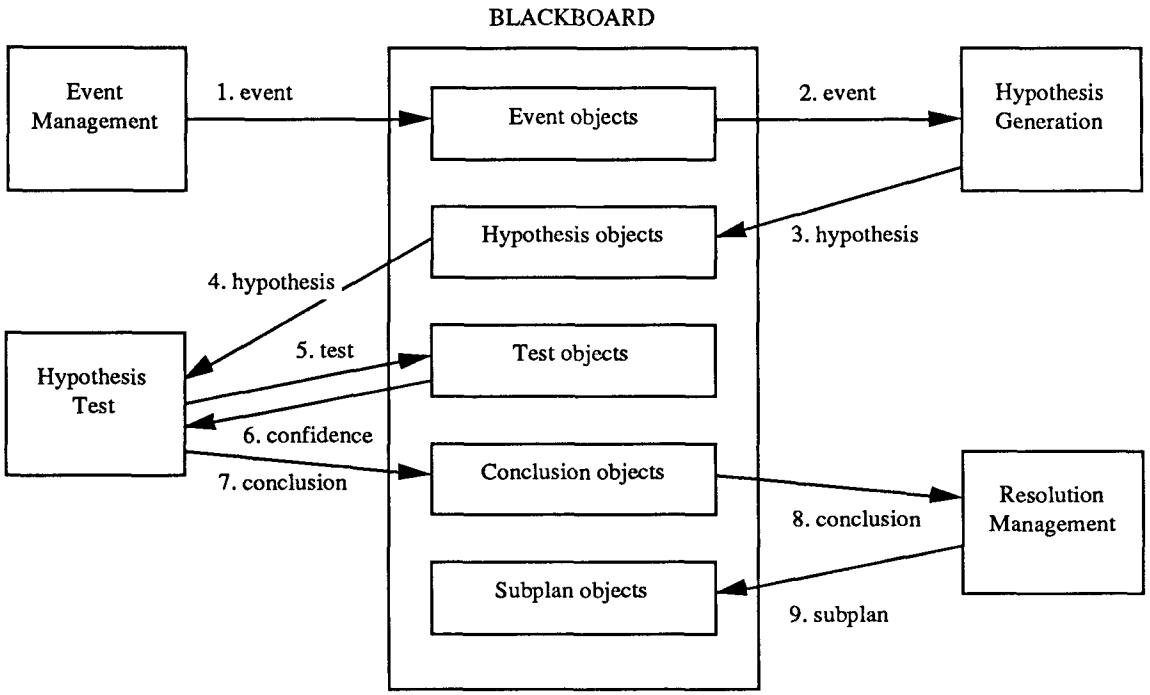

Figure 1. The Blackboard is a global data structure used by the process modules of the NEDS process. The number and label by each arrow pointing toward the blackboard is the sequence in which the objects are created on the blackboard; the arrows pointing away represent information accessed from objects.

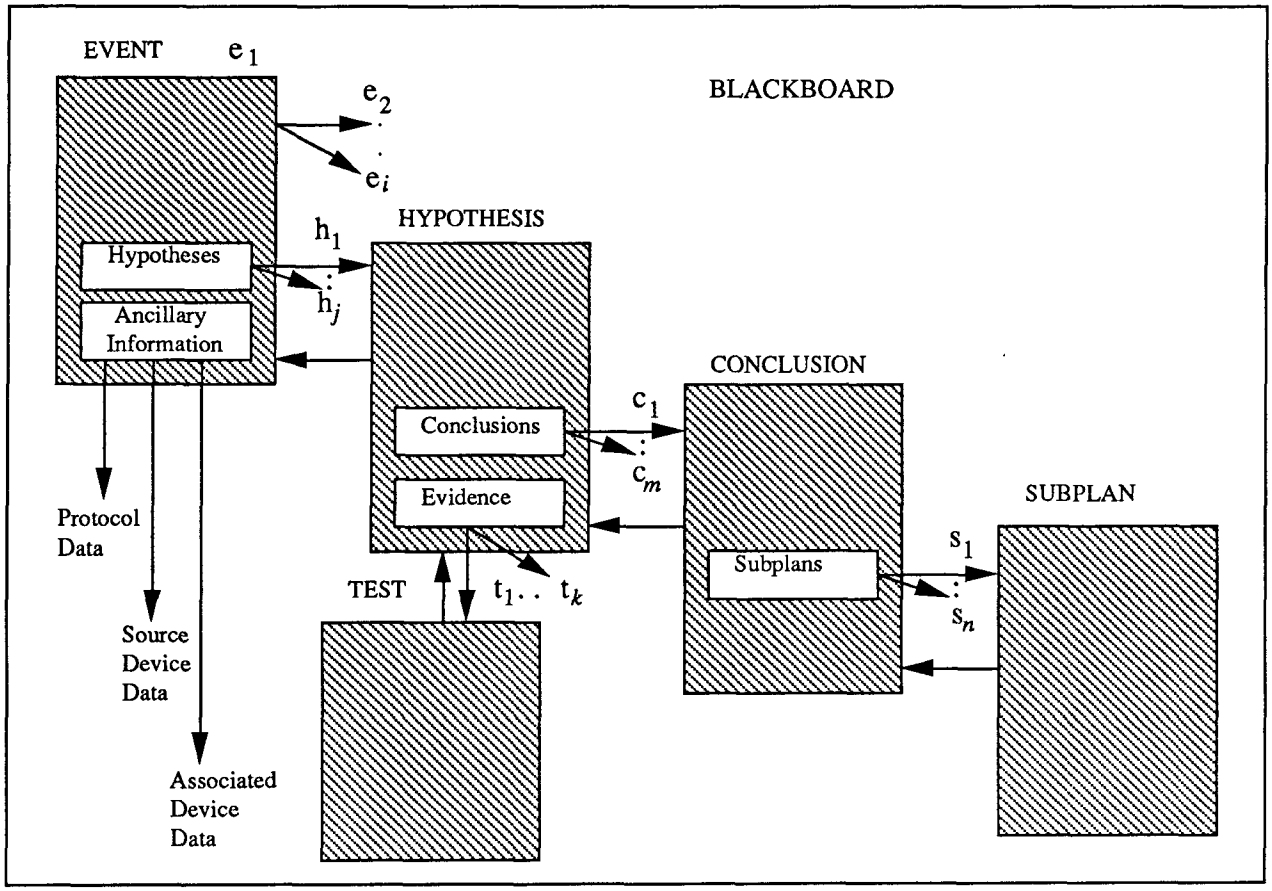

Figure 2. The blackboard contains several types of linked objects (frames). The primary objects correspond to sets E, H, C, T, and S, and are represented as shaded rectangles.

board object independently passes through phases of processing. Second, newly created objects do not produce redundancy. Membership in the set of objects is checked, and if a similar object is already in the set, the new object is coalesced with the existing object. Third, objects in wait state do not require CPU resources, allowing other objects to be processed while waiting for test and command results to be returned from the host network management system.

A metalevel controller schedules four cooperating processes: event management, hypothesis generation, hypothesis test, and resolution management. Each process has a specialized knowledge source. Each blackboard process performs a specialized function. Each blackboard process is independent, and an object scheduled for processing is processed through the next phase. This approach provides near-real-time event correlation and processing of multiple events, and multiple threads of control for self-healing within an internetted data communications network.

A prototype of this system is being implemented in Common LISP on a MicroVAX computer. Common LISP was selected because of its many advantages for symbolic processing. The MicroVAX computer was selected because of a management decision to use the same host computer for the expert system as for the network management system. The same host computer provides all network communication and network management functions required by the expert system.

Performance on the MicroVAX, a concern from the beginning of the effort, has prompted special design considerations. The blackboard processes are independent and, at a conceptual level, operate in parallel. This design characteristic provides an alternative implementation to attain more efficient performance. Hayes-Roth (1985) called attention to the problematic performance of a blackboard architecture expert system implemented in LISP on a sequential (von Neuman) processor. As part of our ongoing effort, we are attempting to transport this design onto a parallel processor. This aspect of the project is discussed further in the last section of this paper.

Each blackboard process has a knowledge source that has been compiled to improve efficiency of search. Each blackboard process knowledge source contains knowledge particular to the process it performs. A separate knowledge acquisition system assists in knowledge acquisition and compilation of the knowledge into its respective knowledge source. (Discussion of the knowledge acquisition system is beyond the scope of this paper).

The blackboard architecture and processes are presented in further detail in the following sections.

4.1 BLACKBOARD ARCHITECTURE

The blackboard, a repository for information objects within a global memory area, is a means for information sharing among processes (Figure 1). Each process performs at least one distinct function and posts its results to the blackboard. There are five information object types: events, hypotheses, tests, conclusions, and subplans (Figure 2). Each object is created in "unprocessed" state and is processed through various phases; each blackboard process advances the object through one particular phase at a time. The numbered arrows in Figure 1 illustrate the general sequence in which objects are created and posted to the blackboard.

Blackboard processes access the blackboard objects through common code functions. The objects are linked together into a bidirectional graph structure that provides direct access to associated objects. For example, event objects are linked to the hypothesis objects that are indicated by the events. Test objects are linked to the hypothesis objects that they support or refute, and so forth. The objects on the blackboard form a current model of fault and anomalous states. The model represents a global view of the entire network, free of redundancy. The model may not reflect the actual network state since it is based on uncertain information. However, uncertainty is managed by computing the degree of confidence using rules within the knowledge sources and representing confidence in the arcs between objects on the blackboard. Degree of confidence is an estimation of the likelihood that the blackboard objects accurately model network state.

The blackboard processes are described in the following sections.

4.2 METALEVEL CONTROL

The metalevel controller is a supervisory knowledge-based process that schedules the execution of other blackboard processes. It processes all input from the host network management system except newly arriving events.

Objects ready for processing are scheduled for processing by the respective blackboard process. An agenda is created by prioritizing the blackboard processes that are scheduled. The highest priority process is executed first.

The blackboard is maintained by the metalevel controller. Information queried by one of the blackboard processes is posted to the appropriate blackboard object by the metalevel controller. Objects that have passed through all phases of processing are archived and purged from the blackboard.

4.3 EVENT MANAGEMENT

The primary function of the event management process is to

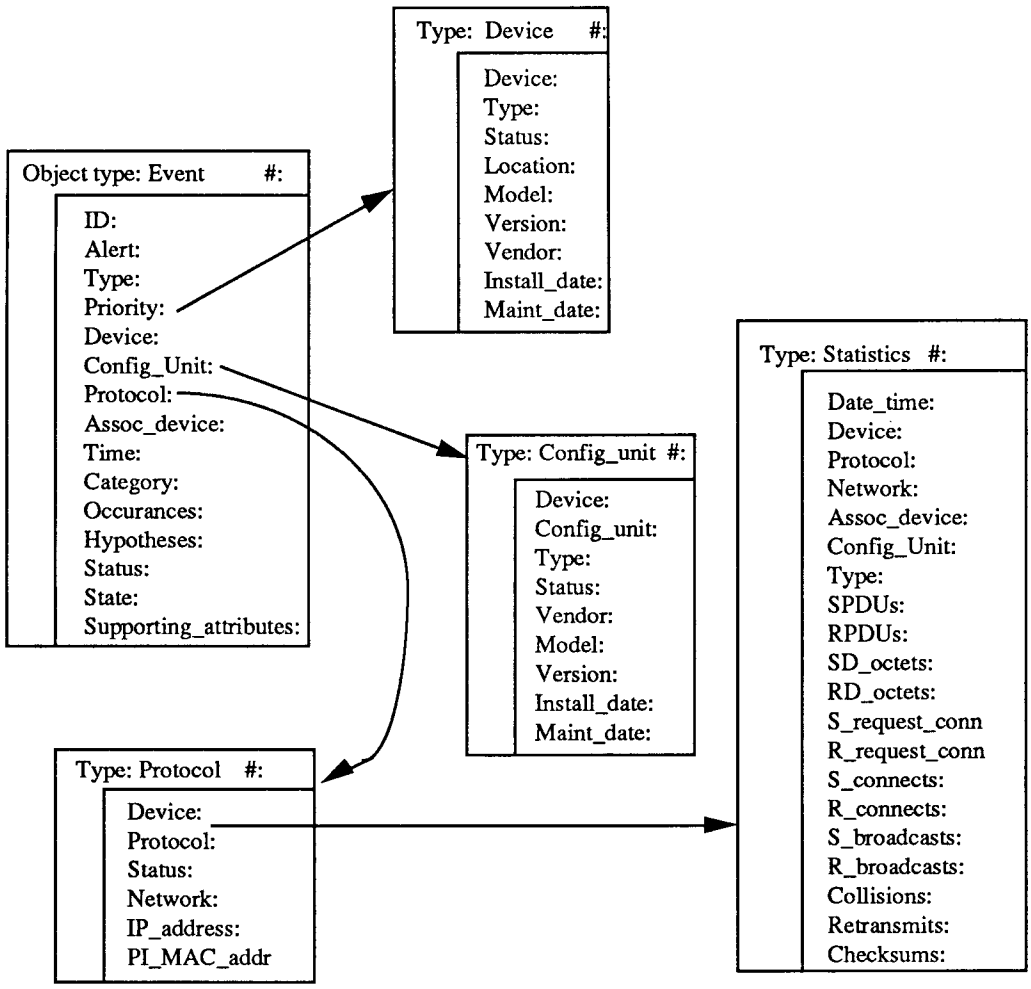

Figure 3. Data is appended to the event object to provide facts about the device that issued the event, the configuration of the device (if known) , the protocol and a snapshot of the associated protocol statistics. Several layers of protocol statistics exist in the database; the statistics that are the best indicators for hypothesis generation are acquired (by rules in the event management knowledge source) and linked to the event.

read new events that have recently arrived for processing. An event is an indication of a fault state or performance anomaly within the network. Unique events are posted to the set of event objects on the blackboard. A pre-compiled knowledge source is used to query the database for network information and statistics, *i.e.*, information that is necessary for processing the event, depending on event type, source, and protocol from which the event was generated.

An event object and associated data are illustrated in Figure 3. There are two primary variations of event objects depending upon the event type. The first is a threshold event that is generated by a threshold violation. This class of event is caused by faults that are either corrected due to retransmission, or are unsuccessful transmissions. In both cases, the occurrences of such events are tallied and stored in the network management database. When a predetermined threshold level is surpassed, an event is generated by the network management system and sent to the expert system for processing. The threshold violation is an indication of an excessive number of otherwise tolerable fault states within the network.

The second type of event is an error report from a specific device within the network. With this type of event an error code is provided with the event message. The error code may or may not pinpoint a specific fault condition within the device, or an associated device, from which it was generated. This type of event reports the relative severity of the fault condition.

In both cases, the information received with the event together with the information obtained from the network management database are posted to the blackboard so that the next process can generate hypotheses as to the specific cause and location of the failure or anomaly.

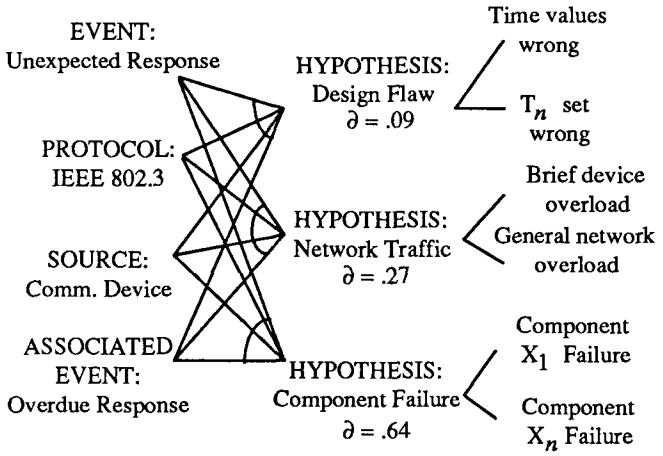

Figure 4. Simplified example of events (left), plausible hypotheses with degree of confidence (∂) and more specific subhypotheses (right).

4.4 HYPOTHESIS GENERATION

A cause of an event can be a number of things from a specific component failure within a device on some network, to complex protocol interactions. A hypothesis is a plausible explanation for a specific fault or performance anomaly. The goal of hypothesis generation is to generate the set of plausible hypotheses that are as specific as possible based on the information available. Available information is limited to that which arrives with the event, is in the network management system database or, in some isolated cases, acquired by query of the network operator. Available information is also considered to be that information accessible within a limited timeframe; information not accessible by the expert system within the allotted timeframe is defaulted to "unknown" and processing continues without it. Event correlation among events in the set of event objects provides indication of faults and anomalies that cannot be identified by the event alone.

Events that are pending additional information are not processed until the information arrives and is posted to the blackboard. When information is pending, hypothesis generation continues for other events. When the hypothesis generation process is out of work (*i.e.*, there are no more unprocessed events on the blackboard), or surpasses a time limit for execution, the hypothesis generation process passes control back to the metalevel controller. Figure 4 illustrates events and associated plausible hypotheses.

Object type: Test # <object number>

ID: <symbol "T" and number>
Type: {Device | DB | OW}
Name: <test name>
Time: <date_time>
Control Level: {0..100}
Parameter Query: <formal command>
Command Parameters: <ordered list>
Command: <formal ASN.1 command>
Result: <symbol>
Confidence: {0..100}
Evaluators (list of 4 tuples):
<{= | > | < } , symbol , conclusion , ∂>
Hypotheses: List of tuples
<object address, ∂>
Parents: <address list>
Siblings: <address list>
Status: {unprocessed | in-process 1|
in-process 2 | processed}
State: {on-hold | interrupt | schedule}

Figure 5. The Test object type and frame structure.

4.5 HYPOTHESIS TEST

The hypothesis test process searches its knowledge source for tests that can provide evidence that supports or refutes each of the hypotheses on the blackboard. Hypothesis objects that are ready (*i.e.*, flagged) for hypothesis generation are processed. For the set of hypotheses being processed, the hypothesis test process creates new test objects, or uses existing tests that have test results that are within age (elapsed time) limitations. The test object type is illustrated in Figure 5.

The set of tests posted to the blackboard are refined to avoid redundancy. If more than one test is found in the knowledge source, an attempt is made to select the one with the most efficient cost. The test with the most efficient cost is the test that provides the greatest confidence in support of the hypothesis at the least cost. Cost is measured by an estimation of the load that the test will place on network resources and the time required to get the result.

Tests form a command string that is sent to the host network management system for execution. Operation on the test object is interrupted until the metalevel controller posts test results on the originating test object. The test object is then scheduled for test result evaluation.

Subsequent hypothesis test process execution evaluates the result and forms one or more conclusions about one or more hypotheses supported by the test. When a threshold degree of confidence in a hypothesis is surpassed, a conclusion object is created and posted on the blackboard. In most cases a hypothesis states a specific failure or performance anomaly; only one specific conclusion can result from one specific hypothesis. However, an event with more than one hypothesis that surpasses level of confidence will likewise have more than one conclusion, (*i.e.*, belief that a network failure or anomaly has occurred at a specific location). The conclusion

with the greatest confidence is activated first, *i.e.*, scheduled for processing. The set of conclusion objects is processed by the following blackboard process.

4.6 RESOLUTION MANAGEMENT

Once the cause of a network problem has been inferred with a reasonable degree of confidence, the problem must be corrected. This is accomplished by generating a plan consisting of a sequence of actions that must be performed on the network. For each conclusion formed about an anomaly in network state, the knowledge source is searched to build a subplan capable of improving that network state. Subplans provide knowledge of how to restore network or device states to normal.

The goal of a subplan is to change a specific state of the network to a desirable state. Subplans focus on a specific location in the network: when possible a subplan focuses on a specific component within a specific device on the network.

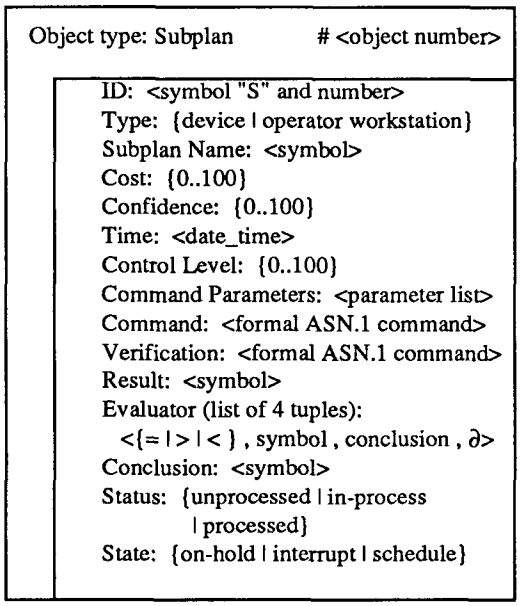

Figure 6 The Subplan object type and frame structure.

Network state change is accomplished through a formal command contained within the subplan object. The semantic is a generalization of an actual command that can be sent over a data communications network to the appropriate device and component. The level of generalization depends upon the device type and protocol level for which it is being issued. The devices that can change network state by on-line commands are a subset of all devices on the network.

To apply the subplan to a specific devicein the network, the subplans are refined with the addresses of specific devices within the network to which the subplan is being applied. A subplan object type is illustrated in Figure 6.

When physical device failure is concluded as the cause of one or more events, NEDS/RC first attempts to circumvent the problem and then sends a trouble report to an engineering activity for fault correction.

5. PERFORMANCE

Expert management systems, to be effective, must perform uninterrupted processing of events at a rate greater than the arrival rate of events. Real-time processing is not necessary because the host network management system queues events for processing, and queues messages from the network until processing is scheduled. However, near real-time performance can be achieved. Processing is performed continuously on the blackboard without interruption. Processing is focused on objects scheduled for a function to be performed. Objects that can not be processed because of priorities or delays due to test or command execution are placed in "interrupt" status until processing can resume. Each object on the blackboard requires one or more phases of processing. When all processing on a specific object is complete the object becomes inactive.

A time slice of NEDS/RC processing is illustrated Figure 7. This illustration conveys an example of the dynamics of the blackboard processes over time. The time line attempts to show several key properties of the blackboard system. First, at time t the hypothesis generation (HG) process is executing. At time t+1 control is passed back to the metalevel controller (MC), which schedules and executes the hypothesis test process (HT). Midway into the hypothesis test process a test message (T_1) is sent to the host network management system. The hypothesis test process continues with other processing until control is passed back to the metalevel controller. The metalevel controller schedules and executes the next highest priority process: resolution management. At time t+3 the result from test T_1 arrives at the network management system which writes the result message to a NEDS/RC input queue (Q). The test results remain in the queue until the resolution management process passes control back to the metalevel controller. The metalevel controller processes all input queues by priority and posts the test result for test T_1 on the originating object on the blackboard and places that object in "schedule" state. The metalevel controller schedules the next highest priority process. The hypothesis test process processes all scheduled test objects. The results of test T_1 are evaluated and stored to provide supporting evidence to one or more hypotheses to which it is linked.

This example, although not fully representative of NEDS/RC processing, does illustrate how NEDS/RC is able to perform in near-real-time. NEDS/RC implementation is currently sequential. The following section discusses the future direction of our research to provide a much more efficient parallel implementation of the blackboard processes.

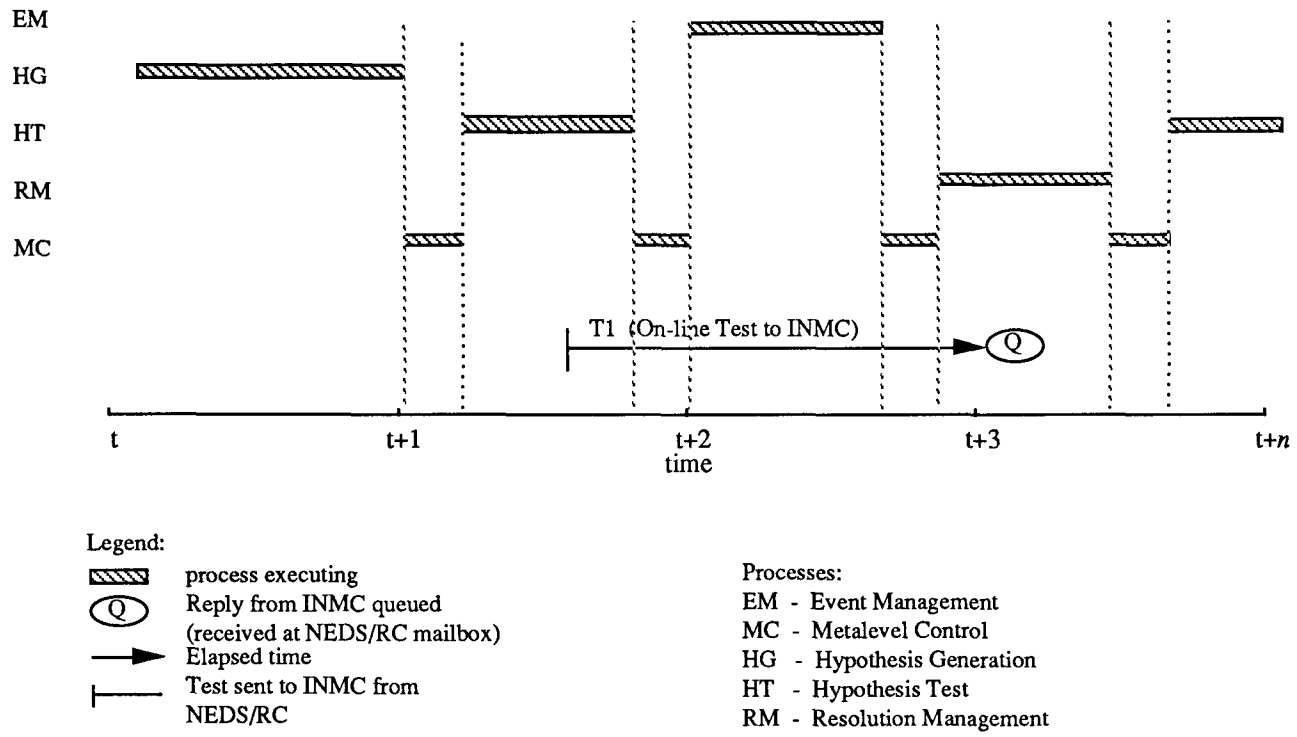

Figure 7. Example of NEDS/RC event processing over time *t*.

6. FUTURE DIRECTIONS

A major objective of this research has been to provide near-real-time event processing for fault and performance management of large data communications networks. An expert manager system needs to perform self-healing within a data communications network as quickly as possible; at a minimum, the average rate of event processing must be greater than the average arrival rate of events. Some communications network environments, such as data communications in hostile environments, mandate fault diagnosis and adaptation to be as fast as possible (Diamond, 1987).

The performance of the current implementation is limited because the average number of computer cycles required to process each event is very large. Performance has been considered in the design: several major processes of the NEDS/RC system can be performed in parallel.

A future direction of this research is to transport the existing Common LISP code now functioning on a MicroVAX to a parallel processing computer. Several parallel architectures have been reviewed. The architecture selected for implementation of NEDS/RC is the multiple instruction multiple data (MIMD) architecture. This architecture will allow parallel processing of the blackboard access functions and all blackboard processes. Within each of the blackboard processes further parallel processing can be performed by additional processors.

Our aim for the near future is to implement our Common LISP code on the AMETEK 2010 parallel processor, a MIMD architecture computer. We expect to improve event processing throughput by an order of magnitude. Simulation of parallel processing of a blackboard architecture system, the Hearsay-II system, has been reported (Fennell, 1977), although there are no clear results on the effectiveness of parallel processing of a blackboard architecture system for real-time fault diagnosis and control.

Within the blackboard processes, further parallelization is possible. These areas for parallelization include: (1) generation of hypotheses by parallel search for plausible hypotheses using a parallel implementation of the Rete algorithm; (2) parallel search for tests that can support or refute hypotheses; and (3) parallel planning, by building subplans in parallel. These claims are made in light of recent research. Examples are data-driven chaining using the Rete algorithm implemented in parallel on MIMD architecture computers (Forgy, *et al* 1984; Gupta, 1987; Gupta, *et al* 1988), and parallel logic programming languages such as PARLOG (Clark, *et al* 1986].

ACKNOWLEGEMENTS

I would like to thank everyone who contributed to this research: Carolyn Stokes for general assistance with the research and for numerous hours spent programming in Common LISP; Maral Achikian for helping with the Common LISP coding; Layli Amiri for knowledge engineering; Lucian Russell, Chris Lightfoot, and especially Jerry Shelton, for many invaluable discussions on a wide variety of topics and for their review of this paper; Steve Smith and Will Copper for many long hours spent in integrating the design of NEDS/RC with the design of the Integrated Network Management Control system; and Hassan Dastvar for his support of this research.

REFERENCES

Bandler, J., Salama, A., "Fault Diagnosis of Analog Circuits," *Proc. IEEE*, 73(8), pp. 1279-1325, Aug. 1985.

Bernstein, L., Yuhas, C., "Expert Systems in Network Management – The Second Revolution," *IEEE Sel. Areas Comm.*, 6(5), pp. 784-87, June 1988.

Boyd, R., Johnston, A., "Network Operations and Management in a Multi-Vendor Environment," *IEEE Communications*, 25(7), pp. 40-47, July 1987.

Clark, K., Gregory, S., "PARLOG: Parallel Programming in Logic," *ACM TOPLAS*, 8(1), pp. 1-49, 1986.

Cantone, R., Lander, W., Marrone, M., Gaynor, M., "IN-ATE/2: Interpretating High-Level Fault Modes," *First Conf. on Artificial Intelligence Applications*, pp. 470-475, Dec. 1984.

Craig, I., "The Ariadne-1 Blackboard System," *Computer Journal*, 29(3), pp. 235-240, 1986.

Cronk, R., Callahan, P., Bernstein, L. "Rule-Based Expert Systems for Network Management and Operations: An Introduction", *IEEE Network*, 2(5), pp. 7-21, 1988.

Diamond, F., "Intelligent Communications," MILCOM-87, *Proc. IEEE Military Comm. Conf.*, Washington, D.C., vol. 2, pp. 745-6, Oct. 1987.

Forgy, C., "Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Matching Problem," *Artificial Intelligence*, Menlo Park, vol. 19, pp. 17-37, 1982.

Forgy, C., Gupta, A., Newell, A., Wendig, R., "Initial Assessment of Architectures for Production Systems," AAAI-84, *Proc. of the Third National Conf. on Artificial Intelligence*, Austin, TX, pp. 116-120, 1984.

Fennell, R., Lesser, V., "Parallelism in Artificial Intelligence Problem Solving: A Case Study of Hearsay II," *IEEE Trans. Computers*, C-26(2), pp. 98-111, Feb. 1977.

Fredman, N., "An Analysis of Qualitative Reasoning and Dependence Modeling in Fault Diagnosis and Testability Assessment," *Third Annual Expert Systems in Government Conference*, IEEE Comp. Soc. Press, pp. 169-75, Oct.1987.

Ganesan, K., Ganti, M., "A Multimedia Front-End for an Expert Network Management System,"*IEEE Sel. Areas Comm.*, 6(5), pp. 788-91, June 1988.

Gilkes, A., Nekhom, A., "Applying Expert Systems to On-line Fault Isolation," *Texas Instruments Engineering Journal*, pp. 19-24, Jan.-Feb. 1986,

Gupta, A., *Parallelism in Production Systems*, Morgan Kaufman Publishers, Los Altos, CA, 1987.

Gupta, A., Tambe, M., Kalp, D., Forgy, C., Newell, A., "Parallel Implementations of OPS5 on the Encore Multiprocessor: Results and Analysis," *International Journal Parallel Programming*, 1988.

Hayes-Roth, B., "A Blackboard Architecture for Control," *Artificial Intelligence*, 26(3), pp. 251-321, 1985.

ISO #9595/2, *Common Management Information Service Definition*, International Standards Organization, New York, 1987.

ISO #9596/2, *Common Management Information Protocol Specification*, International Standards Organization, New York, 1987.

Laffey, T., Perkins, W., Nguyen, T., "Reasoning about Fault Diagnosis with LES," *First Conference on Artificial Intelligence Applications*, IEEE Computer Society Press, New York, pp. 267-73, Dec. 1984.

Merry, M., "APEX 3: An Expert System Shell for Fault Diagnosis," *Journal of Research*, 1(1), pp. 39-47, 1983.

Sutter, M., Zeldin, P., "Designing Expert Systems for Real-time Diagnosis of Self-Correcting Networks," *IEEE Network*, 2(5), pp. 43-51, September 1988.

Ward, T., Bye, P., "Automated Network Management," *Telecommunications*, 19(1), pp. 47-52, January 1985.