DANTES

An Expert System for Real-Time Network Troubleshooting

Robert Mathonet, Herwig Van Cotthem and Leon Vanryckeghem

CI G - Network Systems and Services
Brand Whitlock 87
1040 BRUSSELS - BELGIUM

ABSTRACT

Today's computer networks are large and complex. Their day-to-day operation and maintenance can benefit from the support of an expert system, mainly as an aid in troubleshooting. Network troubleshooting has characteristics, like incomplete data, high rate of events, simultaneous presence of several problems, which raise interesting problems in the development of an expert system.

DANTES is an expert system designed to provide real-time assistance to network operators. This paper presents the system and stresses the development issues that are peculiar to network troubleshooting. Of particular importance are performance of inference in real-time, multi-problem handling, and consideration of time in reasoning and revision of belief Dealing with such issues and especially with real-time efficiency is primarily a question of system design. This has implications for the knowledge base organization, reasoning mechanism, and recording of deductions.

1. INTRODUCTION

DANTES is a real-time assistant to network supervisors in carrying out their troubleshooting activities. Troubleshooting is the part of network control that concerns the detection and diagnosis of network problems with the aim to identify the deficient component. DANTES is driven by external events, which can be spontaneous network alarms, result of tests, manual operator input, etc.
The system analyses each event and, when it detects a malfunction, warns the supervisor, giving advice and indications for the (manual) tasks that need to be performed.

At first sight, network troubleshooting is quite similar to other applications of AI to industrial environment. However, it presents characteristics that make the development of an expert system in this domain an interesting challenge. Of particular importance are :

- Integration of structural and heuristic knowledge:
  DANTES uses structural knowledge, about the network and about network events, and heuristic knowledge about fault detection and diagnosis. Both kinds of knowledge require different representation paradigms, and must be integrated.

- Inference mechanism with real-time efficiency :
  Troubleshooting must be done in real-time.

- Multi-problem handling:
  In contrast with most industrial processes, which only present one problem at a time, coexistence of several problems is standard in large computer networks, and cannot be ignored.

- Reasoning with time :
  In common with other industrial processes 19], time aspects are important in network troubleshooting. They are not limited to correlation of events having occurred at different moments in time. Of special interest is the consideration of time in plausible reasoning [7] and belief revision with time : some deductions must be revised after a time lapse.

* This research was supported by IWONLIRSIA

The purpose of this paper is to discuss the issues that arise from dealing with such features in expert system development. Two aspects are considered in detail : knowledge representation and reasoning mechanism design.

DANTES principles have been tested on a nationwide meshed network interconnecting more than 1000 processors.

The DANTES prototype system has been implemented on a TI Explorer LISP Machine.

2. DOMAIN CHARACTERISTICS

The supervision of a large computer network is usually handled from a control center. Exception events occurring in the network arc reported to the control center through the intermediary of the network itself. These events must be interpreted by human operators to detect and diagnose problems. In this context, a problem is any abnormal situation serious enough to disturb communication between network components. The complexity of troubleshooting is due to the following aspects :

Event Correlation

A single event is usually not significant. An abnormal situation often generates a large number of events, each containing a small piece of information. Situation analysis and characterization must be done by a reasoning process integrating the various events. At first glance, event correlation seems quite straightforward : the reported events need only be divided into groups of related components; the resulting event sequences will then be compared with the standard sequences characterizing a problem and hence will lead us to conclude about the problem. This is not so for a number of reasons:

- *Time Correlation* : Events related to a given situation are scattered over a time interval which can be long : hours, even days for some degradation problems.
  Also, the rate at which events appear can have an importance; e.g. a modem fluctuation once an hour is not serious, but 10 times a minute is.

- *Space Correlation* : An abnormal situation can induce events on several network components : on the faulty component, on hierarchically related components and on components interconnected through the network. A problem must sometimes be indirectly detected, from related events reported by components other than the faulty one.

- *Redundant Events* : Many events are a direct consequence of others and give no additional information. Events can even be reported (via the network) after the problem has been fixed. Such events can only distract the operators and should be masked.

To cover the various aspects of event correlation, especially event space correlation, DANTES needs structural knowledge

* about the network, including its components and the relations among them.
* about events which can occur for these components.

Reasoning With Incomplete Data
Network troubleshooting always begins and often proceeds with incomplete information. Information is gathered incrementally as more events are reported. Event interpretation does not always lead to a definite conclusion : events can be lost or not reported because communication to the control center is down, and different problems sometimes start off with the same event stream. In most cases, only hypotheses can be assumed about a given situation. When incoming events add information, these hypotheses need to be reconsidered or refined. Hence, the inference mechanism must have plausible reasoning allowing revision of belief [7]. Revision of belief is not only triggered by events, but also by time progression. The absence of events can sometimes be as significant as their appearance and should be interpreted as such. For instance, suppose events lead us to suspect some degradation on a component Other events should occur a short while after to confirm the degradation. If no event is reported, the assumption of degradation must be removed.

Molti Problem Handling
Any computer network usually has several unrelated problems pending. Corresponding events generated for each will therefore be intermixed. Some abnormal situations are more critical than others, and should be dealt with first. Since information is progressively gathered, the reasoning related to a given problem is achieved in several stages, discontinuous in time and triggered by event arrivals. In between the stages, the reasoning activity can focus on other abnormal situations. At each stage, the inference mechanism must record all deductions already made to enable the reasoning to continue properly.

Event Explosion
The number of events increases rapidly with the size of the network, typically like N(N-1) where N is the number of active connection points. Our sample network presents an event every 10 seconds on average, sometimes peaking to one per second. This situation adds to the efficiency requirement for the reasoning mechanism, as the system is meant to work in real-time.

3. KNOWLEDGE REPRESENTATION

Three kinds of knowledge must be considered:
* STRUCTURAL KNOWLEDGE;
* DEDUCTIONS, the data types created and manipulated during reasoning activities;
* KNOWLEDGE ABOUT PROBLEM DETECTION AND DIAGNOSIS which specifies how to interpret network events, how to recognize problem situations and how to isolate faulty components.

We believe no single paradigm is appropriate. We have thus combined different knowledge representation techniques.

3.1. STRUCTURAL KNOWLEDGE

Structural knowledge about the network and the network events lends itself to a hierarchical organization that permits inheritance of properties.

A structured object formalism [8] is best suited to represent this knowledge. Each type of network object and of network event is represented by a class in a class inheritance hierarchy. A child class is considered to be a specialization of the parent The child inherits the parent properties but can add to or change them.

DANTES' structural knowledge comprises two basic hierarchies : the network component hierarchy and the network event hierarchy.

Some important object type properties will be used later and are worth mentioning:

a) Network components:
* properties representing relationships between objects (such as the component/subcomponent relationship or the relationship between objects which are physically interconnected);
* properties used by the reasoning process like STATUS, which is a summary of object current situation. Strictly speaking, such properties do not belong to the structural knowledge but are most naturally incorporated in the classes.

The network configuration is represented by instances of object classes as defined above.

b) Network events:
The main properties serve to identify an event in time and space (SEND-TIME. RECEIVE-TIME, SEND-MACHINE, ...) and to define event treatment characteristics (e.g. THRESHOLD and INTERVAL, used in "if the event occurs more than THRESHOLD times during a time INTERVAL, then the situation is serious").

3.2. DEDUCTIONS

3.2.1. GENERAL PRINCIPLES

Deductions can belong to three basic types : SYMPTOM, HYPOTHESIS and RESULT. A symptom represents a (set of) events which may be required for future task handling. Symptom and event differ in their utilization and in their temporal interpretation. An event occurs at a specific moment in time, then it disappears. A symptom is recorded for future utilization in the reasoning process. Symptoms range over time. As such, they have a START-TIME and a STOP-TIME property. This enables symptoms to represent a sequence of events of a same type and having occurred during a time interval.

The distinction between HYPOTHESIS and RESULT lies in their logical interpretation. A hypothesis is an assertion about a network object which may be true or not. A result is a true assertion about a network object. Hypotheses are introduced to handle plausible reasoning in an approach very similar in spirit to [9].

The definition of these new notions forces us to introduce three additional hierarchies. However, an event and its corresponding symptom can be interpreted as different views of the same concept The same is true for hypothesis and result. Different views of a same concept can share identical properties. Using pure hierarchies leads to an inconvenient duplication of knowledge : a same concept can be present in several hierarchies. We have used another solution similar to the viewpoint approach in knowledge organization [2,11] and inspired by [5]. Hierarchies containing identical concepts arc fused in a single lattice. A concept present in several hierarchies is represented in this lattice by a single node containing the properties common to all concept views. Properties specific to a view define a viewpoint of this concept

Deductions are linked by a N to M relationship (DEDUCTION-USED/ DEDUCED-FROM) expressing their use in the reasoning process. The deductions relative to an abnormal situation form a network representing our knowledge about the situation.

3.2.2. MANAGEMENT OF DEDUCTIONS

As a consequence of multi-problem handling, the reasoning process creates and uses many unrelated deductions. These could all be recorded in an ad hoc data structure disconnected from the network objects, a sort of large working memory. This solution does not allow an easy and efficient selection of the elements relevant to the reasoning process. Moreover, it is unnatural since deductions can be associated with network objects : they always represent an assertion about some specific object, and can be recorded there, in a data structure called HISTORY. The first strategy that comes to mind, scattering deduction recording over all the objects involved, has the same disadvantages. The optimal approach consists in associating a history with only some selected object types chosen according to the following criteria:
- the component function in the network;
- the component importance in the network structure : some components divide the network into independent functional units;
- the component size : a component with a large number of subcomponents can record historic data for all its sub- components.

History can contain any number of deductions. Deductions which are no longer valid must be removed. At first glance, all deductions related to a problem could be removed when a conclusion is reached. However, recording of conclusions is particulary useful for a direct diagnosis of problems which occur repetitively on the same network component Say, if a connection has presented failures due to a faulty subcomponent (a modem for instance), the next time a failure occurs for this connection, one can likely suspect this subcomponent Several mechanisms can be employed for the selective garbage collection of history : DEDUCTION FIXING and TIMEOUT features. They are discussed in section 4.

3.3. PROBLEM DETECTION AND DIAGNOSIS KNOWLEDGE

Recent applications of knowledge based techniques to industrial systems have introduced the concept of deep reasoning [1] : the system is represented by a deep model which integrates a representation of the system structure with its components and the relations among them, and the functional description of each component [4,6]; reasoning is carried out by simulating system functioning from this model.
This technique does not apply here because the knowledge is mostly heuristic : a large computer network is very complex; knowledge about the network behaviour is also very fragmented among several experts each having a specific domain area. Simulation is inefficient, often impossible, and in general not very useful. Model based reasoning can only be used at a very elementary level, like in "if a switching node is down, all processors connected to it are not obtainable".
The heuristic nature of the knowledge would lead us naturally to a production rule representation. To meet task requirements, the rules should have the following features :

- They must be well integrated into the structured object formalism used to represent structural knowledge : rules should easily access objects of the structural knowledge and their properties (including history and status).
- The selection and application of rules related to a problem must be very efficient to cope with the real-time constraint.

An alternative is to represent detection and diagnosis knowledge using techniques of procedural attachment [8] to structured objects (demons in frame terminology [2]]. Event treatment demons are defined for network object types. When an event occurs, the corresponding demon is executed. This technique is certainly efficient but not flexible.

The representation technique we have chosen combines the advantages of rules and demons. In our approach, a rule is defined on an object class and rule application takes place for an instance of this class in a way very similar to the LOOPS rule oriented programming approach (10]. Rule definition has the following properties :

- *Class* : The class to which the rule is associated. For DANTES domain, it references an object type in network representation. Rules are class properties inherited in the classical way.

- *Name* : The rule name.

- *Trigger* : It specifies the network events or system actions which can cause rule application. Typical actions considered as trigger are deduction assertions made by the system. The trigger property can then reference the precise viewpoint which may trigger the rule, or simply the whole concept type if concept view is not important for rule triggering.

- *State* : The status value that a network object instance must have to enable rule application,

- *Environment*: It allows the declaration of variables local to the rule. This declaration is optional. Note that a rule environment always include some local variables automatically bound by the system (e.g. CURRENT-OBJECT bound to the instance for which the rule is invoked).

-*Body* : The body contains any LISP forms, to be evaluated sequentially. Generally, the body is an (IF condition THEN action) form where condition and action concern the history of network objects known in rule environment.

Example

```
(DEFRULE (CONNECTION :R1)
  (TRIGGER event modem-status-change)
  (STATE (up degrading))
  (IF
     (Is-suggested current-object 'degradation)
     (THEN (suggest current-object 'result:failure))))
```

This rule, with name :R1, is defined for class CONNECTION. The rule is triggered by the EVENT MODEM-STATUS-CHANGE (we specify the viewpoint to consider) on CONNECTION instances with STATUS set to UP or DEGRADING. The rule body states that if the current object already presents a degradation (hypothesis or result without precision), then one can conclude that a FAILURE exists on this connection and one specifies the viewpoint of the FAILURE : it is a RESULT (the keyword RESULT: prefixes FAILURE).

With these rules, problem detection and diagnosis expertise can be expressed succintly, in a declarative way. Moreover, the rule base is not flat : rules are grouped by object class. This organization allows a distribution of expertise among the different object types of the network representation at the very place where the application of knowledge can be useful and efficient. Rule organization also helps to efficiently select the rules related to a given problem. As rule application takes place on an instance of a class, the selection of the rules which can be applied can directly start from the subset of rules defined for this class. The trigger and state properties allow a flexible and powerful control of rule selection and application. Both attributes are used for a precise description of the situations where the rule may be applied.

4. REASONING PROCESS

4.1. GENERAL PRINCIPLES

Events occurring in the network are transformed into DANTES internal formalism (object oriented). The network objects concerned by this event are determined from information associated with each event type. A message reporting the event is sent to each object. At message reception, the object selects, from the rules associated to it, those whose trigger matches the event received and whose state matches the object's status value. The selected rules arc tried sequentially in their order of selection. This rule application can produce deductions which, in turn, can trigger other rules, thus continuing the inference, and possibly updating the deduction network. Figure 1 illustrates this principle.

This inference mechanism restricts problem solving activities to only the network objects concerned by the problem. Together with the corresponding distribution of history and the organization of rules, it makes the inference process manipulate only knowledge relevant to the problem. This permits to handle multi-problem reasoning in an efficient and elegant way.
One can view the logic followed as a generalized "state transition reasoning" as commonly used in communication protocol design. Each object is in a well defined state (given by STATUS). The occurrence of events and the resulting deductions will eventually alter the state of one or several objects.

The inference mechanism comprises three basic operations :

* create deduction;
* remove deduction set and update the deduction network resulting from this deletion ;
* fix a deduction set, i.e. remove from the deduction network related to an abnormal situation, all those deductions which arc not used to deduce the given set.

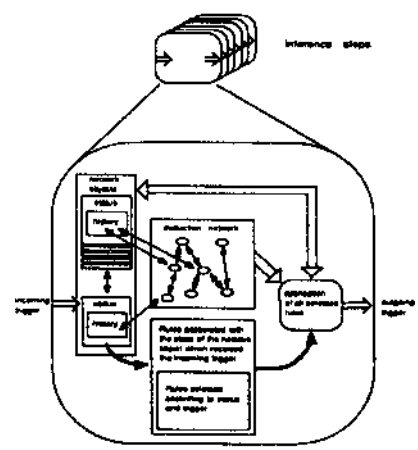

Figure 1

4.2. TIME ASPECTS OF INFERENCE

From section 2, we glean that the two most important aspects related to time are:
- time correlation between events ;
- belief revision with time : some deductions become invalid after a while (i.e. they have a "lifetime"), and should thus be removed. Deciding when a deduction gets too old is part of domain expertise.

These features can be implemented in several ways : integrated in the heart of the system, by functions, or by rules. Knowledge about time correlation is best expressed by rules. Conditions for time correlation of events refer only to the properties of :
- the incoming event;
- symptoms or other deductions to which this event can be related;
- network objects concerned by this event.

On the other hand, revision of deduction with time is directly concerned with the whole deduction network to which they belong. If a deduction is used to derive other (valid) deductions, it cannot be invalid. Checking whether a deduction can be revised, and subsequently removed, could be implemented in the rule base. This is however highly inefficient (these rules have to be tried on a regular time basis) and would pollute the rule base with knowledge which is in fact control knowledge and not domain expertise. As amply stressed in the literature [31, both kinds of knowledge must be separated for the transparency of the knowledge base.

Our approach is as follows : to handle revision of deductions with time, we have added to each trigger object type a timeout viewpoint, which contains all properties allowing time management of these objects in the reasoning process. A deduction is IN TIMEOUT if and only if the deduction has not been IN USE for a predefined time (the timeout interval). A deduction is not IN USE during a certain interval if and only if :
- it has not been used to derive any deduction(s);
- no deduction deduced from it is still IN USE.

The implementation uses a separate timeout process which maintains a list of timeout viewpoints for deductions which can be in timeout. When a deduction times out, the timeout process reports to the associated object.

The timeout viewpoint of a trigger can be used to trigger rules just like the other viewpoints of trigger objects (event, symptom, hypothesis or result). This allows removal of deductions after a certain period and also :
- specific action for deductions at regular times;
- polling of certain conditions by the inference engine.

These facilities allow us to deal with time aspects in expert systems, without increasing the time required for rule selection.

4.3. REALTIME ASPECTS OF INFERENCE

Real-time considerations are important for the design of DANTES. In the following, we discuss several features built in DANTES, which arc typical of traditional real-time systems.

Real-time implies fast response, hence one must:
- minimise the amount of code executed;
- avoid disk accesses by limiting virtual memory size and usage.

The rule selection mechanism, using state and trigger properties of rules, limits drastically the number of rules tried at each inference step. For our sample network, the ratio between the number of rules selected at each inference step and the total number of rules in the knowledge base is at most 1/10.

All aspects concerned with memory management are mostly machine dependent. We have incorporated the following features :
- deductions are allocated and deallocated by the inference engine, in a special area on which no garbage collection is done;
- the structural knowledge is static and can be loaded in a static area (no garbage collection);
- a judicious choice of which parts (code or data) should stay in physical memory leads to a minimization of disk access;
- code reduction and optimization are used in the rule compiler, augmenting the code efficiency;
- the dynamic work area is quite small, with frequent on-line garbage collection.

5. CONCLUSION

This paper introduced DANTES, an expert system to assist network operators in the maintenance and problem diagnosis of a large computer network. Apart from its significant economic value, DANTES presents interesting functional aspects which are not found in other industrial expert system applications :
- It is driven by discrete external events;
- Time aspects are important, not only in the correlation of events, but also in the reasoning process;
- Multi-problem handling;
- The inference process must be real-time and sustain a high rate of incoming events.

DANTES combines a number of paradigms, each well suited for a part of the problem. The salient aspects of the proposed system are :
- the integration of heuristic and structural knowledge;
- the ability to treat several distinct problems concurrently;
- a fast and efficient rule selection and rule application mechanism;
- a fast and original approach to handle time in the inference process;
- various design and implementation issues to provide real-time performance.

We believe that the proposed solution forms a framework that can be used in many real-time expert system. The inference mechanism provides a natural way to deal with time dependent knowledge without imposing undue restrictions on the formulation of the rules.

6. ACKNOWLEDGEMENTS

We are grateful to Francoise Van Den Berghe, our domain expert, for her continuous cooperation throughout this project. We thank Suzanne Galand for her helpful comments about this paper.

REFERENCES

[I] Bobrow, D., and P. Hayes, "Special issue on qualitative reasoning." Artificial intelligence. (1984).
[2] Bobrow, D. and T. Winograd, "An overview of KRL a knowledge representation language." Cognitive Science. 1:1(1977) 3-46.
[3] Clancey, W. "The advantages of abstract control knowledge in expert system design." In Proc. AAAI -83. Washington DC, United States, August, 1983, pp. 74-78.
[4] Davis, R. "Diagnosis reasoning based on structure and behaviour." Artificial intelligence. 24 (1984) 347-410.
[5] Elio, R. and J. de Haan "Knowledge representation in an expert storm forecasting system." In Proc. IJCAI-85. Los Angeles, United States, August, 1985, pp. 400-406.
[6] Gallanti, M., L. Gilardoni, G. Guida and A. Stefanini "Exploiting physical and design knowledge in the diagnosis of complex industrial systems." In Proc. ECAI-86. Brighton, United Kingdom, July, 1986, pp. 335-349.
[7] Hayes-Roth, F., D. A.Waterman and D. Lenat Building expert systems. Addison Wesley, 1983.
[8] Nilsson, N. J. Principles of artificial intelligence. Tioga Palo Alto, 1980.
[9] Paterson, P., P. Sachs and M. Turner "Escort : the application of causal knowledge to real-time process control." In Expert systems 85 Cambridge University Press, 1985. pp. 79-88.
[10] Stefik, M., and D. Bobrow The Loops manual. Xerox Palo Alto Research Center, 1983.
[II] Stefik, M. and D. Bobrow "Object oriented programming : themes and variations" Ai Magazine 6:4 (1986) 40-62.