43

Fault Isolation and Event Correlation for Integrated Fault Management

S. Kätker, M. Paterok
IBM European Networking Center
Vangerowstr. 18, D-69115 Heidelberg, Germany
Tel.: +49 6221 59 4404 / 4312
e-mail: {kaetker,paterok}@vnet.ibm.com

Abstract

An algorithm for fault isolation and event correlation in integrated networks is presented. It reconstructs fault propagation during run-time by exploring relationships between managed objects and provides improved focus and efficiency compared to similar algorithms. The functionality of the algorithm is generic, straightforward, efficient, and applicable for different management architectures such as SNMP and TMN. Clearly defined interfaces allow parallel execution of problem investigation, integration of different management architectures and systems, and incorporation of other correlation techniques.

Prototype and production level systems have been implemented for TMN- and SNMP-based management systems to validate the concept and for use in real networks. This development was done after an extensive study of requirements and existing techniques, the results of which are also presented.

Keywords

Fault Management, Service Management, TMN, Fault Isolation, Event Correlation

1 INTRODUCTION

Aside from the growth in size and speed, telecommunication networks have become more and more complex in their layered structures and services. Upcoming integrated backbones using Asynchronous Transfer Mode (ATM) and Synchronous Digital Hierarchy (SDH) simultaneously carry voice, data, and video traffic, thereby dramatically increasing the number of protocols in use. A typical scenario comprises applications running over TCP/IP which is using ATM which in turn runs on SDH over dark fiber lines. As a result, a broken link in the underlying backbone or any

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

other problem in the base transmission services causes a large number of higher layer services to fail. These services may not even be directly connected to the failing component, i.e. a failing SDH link would disturb operation of the TCP/IP networks and connections including applications in possibly very large numbers.

This type of problem causes hardship to the network management systems: floods of events are generated by the management systems of the various networks. If there is no integrated management system the operators need to correlate the problems on the various network management consoles manually using their experience. This is a daunting task which usually takes too long in the face of high downtime costs, if it is possible at all. An integrated management system, if present, is flooded with events from all the different subnetwork management systems.

A problem which is largely unresolved in practice (and in theory) is that of automated fault isolation and event correlation for integrated networks. *Event correlation* means that all problem indicators (i.e. events) whose generation have been caused by the same underlying problem are grouped together. *Fault isolation* stands for the automated detection of the root problem which is the cause of the problem. In this paper, an algorithm is presented which allows to determine the root cause of network problems and correlate the events associated with the problem. The algorithm requires a limited integration of the management system. It is based on model traversing and allows incorporation of different correlation techniques. The algorithm is currently being implemented and will be used for the management of real-life telecommunication networks. It was patented by IBM (Kätker and Paterok, 1996).

The requirements which an event correlation solution for integrated telecommunications networks should meet are listed in Section 2. The different approaches and techniques known so far are also discussed. The algorithm is presented and explained in Section 3 and aspects of its implementation are described in the following section. A real-life correlation example is shown in Section 5 before the paper ends with the conclusion.

2 REQUIREMENTS AND TECHNIQUES

This sections lists the critical requirements for effective fault localization and event correlation. The existing literature is discussed in light of these requirements. The suitability of these techniques may differ depending on the management layer where it is applied.

2.1 General Requirements

In order to be effective, the technique has to cope with the following general requirements:

- **Integration with Network Management Infrastructure**
  Each correlation algorithm requires the underlying network management sys-

tem to perform specific functions, e.g. information requests. These functions must be easily embeddable into existing network management systems in practical use.

- **Flexibility**
  Networks are dynamic with respect to their structures and components. It must be possible to adapt the correlation system to changes in the network topology, the component types and versions, and the services offered. Implementation of the changes must be straightforward and controllable to keep failures as a result of changes to a minimum level.
- **Performance and Parallelism**
  Algorithms requiring a central component to be contacted upon each correlation step will suffer from performance problems when large numbers of events flow in. Parallelism in this case covers external parallelism where the algorithm can be distributed on several physical machines and internal parallelism which stands for the use of threads on the same machine. Requests for network information usually take much more time than the correlation itself, in particular in faulty networks where information requests are frequently timed out. It is therefore important to perform these requests in parallel by use of threads.
- **Functional Distribution and Combination of Techniques**
  Levels of integration are limited in most network management systems in use now and in the near future. The algorithm must therefore lend itself to distribution over different network management systems with possibly different architectures. These different systems may perform correlation autonomously using different techniques, as long as they comply to clearly defined and straightforward interfaces.
- **Robustness Against Incomplete Data**
  The technique must be able to perform its task in the face of incomplete data, i.e. missing events and timed-out information requests caused by faulty networks.

2.2 Fault Isolation and Event Correlation Techniques

Most known techniques can be classified by one of four different approaches. Subsequently, these approaches are sketched, the key references are mentioned and the advantages and drawbacks are discussed in light of the requirements listed above.

Model Based Reasoning Tools

Many published concepts and implementations use artificial intelligence methods for fault management. To overcome the lack of structure in the rule bases, the second generation of expert systems, called model based reasoning systems (Kehl and Hopfmüller, 1993, Jakobson and Weissmann, 1993), use different techniques to represent structural knowledge (using object-oriented models) and heuristical knowledge (using rules).

A maximum of flexibility and the ease of customization of basic correlation rules are the major advantages of model based reasoning systems. Major disadvantages

include limited performance and lack of parallelism. Customization and maintenance of large rule bases is a critical task. Also, rule based systems frequently suffer from lack of communication with their environment, e.g. for exchange of information. Model based reasoning systems play a very important role in the area of device level fault management and for executing escalation rules or fault management policies, e.g. to determine which trouble tickets have to be created under which circumstances. In this role the expert system serves as an integrator between the different fault management techniques.

Fault Propagation Models

These models describe which symptoms will be observed if a specific fault occurs. Kliger et al. (1995) use the concept of causality graphs to decode the set of observed symptoms. Bouloutas et al. (1992) use intersections of sets of possible failures that may cause a given symptom.

Fault model based approaches require an a priori specification of the fault-causes-symptom relationship for any fault that might happen in the system. This is hard to obtain in practice in particular when the network is subject to change. If the fault-causes-symptom knowledge is easy to determine for a given system, fault propagation model based tools can provide a robust and efficient event correlation solution.

Model Traversing Techniques

These algorithms reconstruct fault propagation during run-time by using relationships between managed objects. Jordaan and Paterok (1993) present an algorithm for event correlation based on object model traversing in a management system which is based on the Telecommunications Management Network (TMN, (Aidarous and Plevyak, 1994)). Starting with the managed object that emitted the trigger event related managed objects are discovered stepwise and their events are correlated. The algorithm presented in this paper is related to the algorithm from (Jordaan and Paterok, 1993) but has clear advantages in the direction and efficiency of fault isolation. The technique described by Jordaan and Paterok is targeted to a specific information model, it provides correlation, but no directed search or fault isolation. Based on an object-oriented network model an event correlation system model for the physical layer of large telecommunication networks is presented by Houck et al. (1995). The service dependency graph introduced in (Kätker, 1996) models a client server application and the network it uses as a set of abstract services and the dependencies between the services. A generic algorithm is presented that traverses the dependency graph to extend fault isolation and event correlation to the application level.

Model traversing techniques work best when managed object relationships are graph-like and easy to obtain. Their strengths are high performance, potential parallelism, and robustness against network change. Their disadvantage is a lack of flexibility, in particular if fault propagation is complex and not well-structured.

Case Based Reasoning Tools

The goal of these tools is to suggest a solution for a given problem by citing a solution that solved a similar problem in the past. Therefore, case based reasoning tools are usually not used for fault isolation, but for the following fault diagnosis task (see Dreo and Lewis (1995) for an example).

This technique analyses the phenomenons of a given problem, i.e. no knowledge about the fault propagation is required. However, a number of similar problems have to be resolved before the user can start to compare past and present data. Incomplete or wrong results may occur if the network configuration is changing. Also, algorithms for finding similar, not necessary equal, problems by comparing symptoms is non-trivial ('fuzzy query').

2.3 Layer-Specific Requirements

The requirements and suitable techniques for fault isolation and event correlation vary significantly depending on the management layer and function where it is applied (see Stallings (1993) for the functions of the various layers).

Network Element Layer

This layer is involved with a variety of different hardware devices which often do not support standard management protocols or object models. Event correlation must therefore deal with incomplete information and support proprietary interaction with devices. Since component behavior upon failure heavily varies depending on device type and manufacturer, the corresponding fault isolation technique should provide a maximum of flexibility, e.g. if a new equipment type or version is installed. Event rates are mostly low to moderate.

These requirements make expert systems a suitable tool for network element layer event correlation, provided that the expert system is tightly integrated with the management system.

Network Layer

In general, any communication network can be treated as a layered graph. More specifically, some graph-based standard models exist (e.g. the ITU M.3100 generic network model (ITU, 1995), which is widely used in the TMN and TINA (Richter et al., 1995) arena) to model various kinds of telecommunication networks. As these network models represent the layered physical and logical structure of a telecommunications network event storms occur specifically in this layer. Since the overall network frequently changes because nodes are added and/or removed, network layer event correlation should explore the network structure during runtime.

That suggests the use of model traversing techniques for network layer event correlation.

Service Layer

Currently no standard models have been defined for the service layer. They would be characterized by a complex, hierarchical structure of objects representing ser-

vices, applications, subcomponents, and the dependencies between them, resulting is some sort of dependency graph (see TINA (Dupuy et al., 1995) for approaches). Because high event rates have to be expected for service layer management, a dependency graph walking algorithm is a suitable technique.

2.4 Combination of Techniques

The above survey shows that there is no single technique dominating for all aspects of real-life network environments. To provide comprehensive fault isolation and event correlation, it is therefore recommended to combine different techniques in a distributed and partly autonomous fashion. In particular, expert systems and model traversing algorithms offer complementary capabilities, the first being good at complex, flexible, unstructured, non-distributed problems, while the latter are generic, high performing, simple, and can be distributed. The current development described in Section 4 is capable of combining these two techniques. A model traversing algorithm identifies the failing network node and correlates possible events from other nodes. An expert system analyses the devices themselves and executes fault management policies by, e.g., setting trouble ticket parameters, issuing rerouting requests, or paging technical support staff.

The remainder of this paper concentrates on the description of a generic model traversing technique for network layer fault isolation.

3 ALGORITHM

The algorithm uses relationships between network hardware and software components and their managed objects to search for the root cause of a problem. This search is carried out using three types of information request functions. To enable the algorithm for a specific network management system only these functions need to be implemented. The algorithm can therefore easily be adapted to different networks and management systems.

Assume that a connection A→Z from node A to node Z fails and an event is generated as a result. This event triggers the algorithm which performs the following steps:

Horizontal Search

Starting from node A, the next component (node) along the path of the connection A→Z is identified and the current operational state of this node and the connecting link is determined. This is repeated for every hop along the path until one such component is non-operational.

If all components are operational, then no problem could be detected for the trigger event and the event is declined or passed on for manual treatment. If a problem is identified, it replaces the current problem as the root cause of the problem. If the root cause has already been detected as a result of a problem search triggered by another event, event correlation takes place, i.e. the second event detecting the problem is appended to the first one and further problem search

is performed for the first problem only. Horizontal search is carried out by continued application of the following two functions for information request:

`getNextHop(A → Z, B)`	get the node representing the next instance downstream from B that is used for the communication between A and Z.
`getOperationalState(A)`	get the operational state of A.

Vertical Search

Once horizontal search has detected a component as being faulty it is determined whether this component is an elementary object (e.g. a repeater) or whether it consists of multiple components (e.g. a link using a path or connection in an underlying network). If the component is composed the problem search is carried one level lower to be continued at the underlying network level. This is supported by the information request **`getLowerLayerSAP`**, which determines the underlying path's start and end points. Horizontal search is then performed for the path between these two points.

`getLowerLayerSAP(A)`	get the node representing the lower layer Service Access Point (SAP) that is used by A.

Termination of Search

A search terminates when one of the following applies:

- A non-operational component was identified by the horizontal search algorithm. If this component is elementary, i.e. the search cannot proceed in an underlying network, the problem is reported to the operator.
- If the horizontal search algorithm cannot detect a non-operational component in the path the problem is declined and either purged or reported to the operator for manual treatment.
- The root cause of the problem was identified and is identical to the root cause of another open problem. The problem is appended to the other problem, search stops for the appended problem and may be continued for the other problem.

The overall outline of the algorithm is as follows: horizontal search is applied until a faulty component is detected. If this component is composed, vertical search is applied for the start and end points of the failing component to carry the search to the next lower level. Then horizontal search starts for the lower level. All this is repeated until a termination rule applies, e.g. the failing component is elementary.

Figure 1 explains how the fault isolation algorithm works for an example network by subsequently applying the three information request functions. The scenario is an TCP/IP network running over a multi-segment LAN. A LAN segment fails and causes a TCP connection to abort. The corresponding TCP event triggers the start of the algorithm. Horizontal search using **`getNextHop`** and **`getOperationalState`** verifies that connectivity at the TCP layer is lost. To further explore the problem in the underlying IP network vertical search takes place by applying the function **`getLowerLayerSAP`** to the two end points of the TCP connection. During the following horizontal search, **`getNextHop`** uses the IP routing table to first find out that

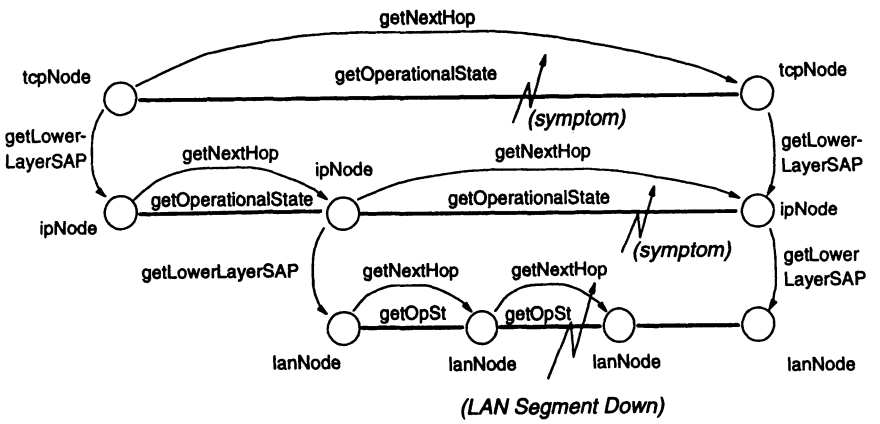

Figure 1 Illustration of the Algorithm for an IP Network Running on a LAN.

the first hop is operational and then detect that the second hop is not. Vertical search carries the search to the next lower layer, the LAN level. Horizontal search then detects that the LAN segment between the two middle lanNodes (which normally represent bridges) is down. The result of `getLowerLayerSAP` reveals that this is an elementary object, i.e. there is no underlying network from a management perspective. Search stops and the result is communicated to the operators.

The algorithm will be further illustrated by the implementation aspects described in Section 4 and in the example given in Section 5 which contains event correlation.

4 IMPLEMENTATION ASPECTS

IBM is developing prototypes and production level systems that implement the algorithm while meeting as many of the general requirements from Section 2 as possible. To support integration with different network management infrastructures the kernel algorithm is separated from the actual information access modules (see Figure 2). The Information Access module implements the three elementary functions `getNextHop`, `getLowerLayerSAP`, and `getOperationalState` for different management systems and techniques without changing the kernel part of the algorithm. Experience from prototypes and product development shows that the three elementary functions can be provided for a variety of standard and proprietary infrastructures. Prior to the implementation described in this paper, prototypes have been developed for TCP/IP networks (based on SNMP, MIB-II) and for Token Ring LANs (based on a proprietary information model). See Stallings (1993) for the Simple Network Management Protocol (SNMP) and MIB-II.

The kernel part drives the fault isolation process, administers the problems including correlation and handles the threading. Internal parallelism is implemented using DCE threads. Each thread executes the algorithm for a given symptom in

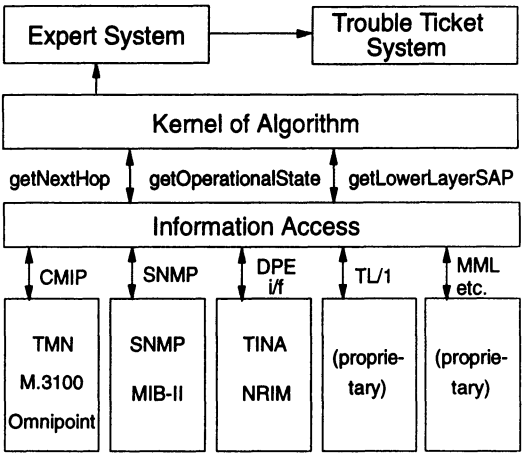

Figure 2 Architecture of the Fault Isolation and Event Correlation System.

parallel resulting either in correlation and termination or in continued search. External parallelism is implemented by migration of threads from one fault isolation and event correlation agent to another. Each agent is responsible for a given management domain or layer. If a thread detects that further diagnosis is needed for an object that is located in another agent, the problem reference and the identification of the object is transferred to the neighboring agent. Since the migration interface is well defined, an interface definition language like CORBA's IDL (OMG, 1995) can be used in order to extend the external distribution over technology and administrative boundaries. Threads may migrate between public network operators or to and from a customer site as well.

Maximum performance is ensured by placing the correlation and isolation process as close to the monitored networks and devices as possible. The distributed algorithm was therefore implemented in the network level agents whenever the architecture allows, e.g. for the TMN-based version using the M.3100 model. If this is not possible, e.g. for the SNMP-based version, the algorithm was implemented in the network manager components using internal parallelism. However, caching of relevant object information can be used to improve performance if needed.

The information access interface was implemented for both the TMN-based M.3100 information model (ITU, 1995) and the SNMP-based MIB-II. In the following, the TMN-based approach is discussed in more detail. In terms of OSI management the information access interface acts as a manager application to the M.3100 network level agent that maintains the managed objects for a given subnetwork.

The managed object classes `trail` and `connection` provide the information for the algorithm as shown in Figure 3. A `trail` represents an entity that is responsible for the integrity of transfer of characteristic information while a `connection` represents an entity that is responsible for the transparent transport of information. `trail` and `connection` contain attributes that point to the `trail/connection` termination points (`a-TPInstance` and `z-TPInstance`) in the corresponding network

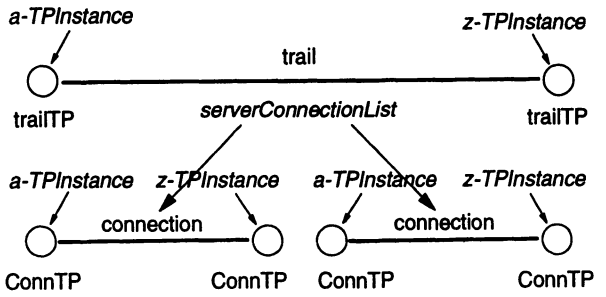

Figure 3 Relationships Provided by the M.3100 Standard Object Model.

elements that are logically or physically connected by the `trail` or `connection`. A `trail` may contain a `server connection list` attribute that points to the connections that serve the `trail`, i.e. are used to build the `trail`. `connections` may contain a `server trail list` attribute that points to the the `trails` that serve this `connection`, i.e. the `trails` that are used to build the `connection`.

The 3 basic functions provided by the information access interface are implemented as follows:

- `getOperationalState(A)`
  get the `operationalState` attribute of the `trail` or `connection` given by `A`.
- `getLowerLayerSAP(A)`
  get the first `connectivity` object in the `serverConnectionList` or `serverTrailList` of the `trail` or `connection` given by `A`.
- `getNextHop(A` $\rightarrow$ `Z, B)`
  get the next `connectivity` object in the `serverTrailList` or `serverConnectionList` of the `connection` or `trail` given by `A` $\rightarrow$ `Z` after the `connectivity` object given by `B`. If `B` is undefined return the first `connectivity` object. If `B` is the last `connectivity` object in the list return undefined.

Implementation of these three functions enabled the generic algorithm to perform on TMN systems using M.3100 for network layer modelling.

5 EXAMPLE

This section illustrates the algorithm and its distributed implementation by a simple example. Consider a client server application running over a TCP Wide Area Network (WAN) connection. The WAN part of the TCP connection is provided by an ATM virtual path as sketched in Figure 4.

The ATM network uses SDH for the physical links between the switches. There are two management domains in this network. The customer running the client server application operates an SNMP management domain responsible for the management of his IP network. The customer is using a virtual private network (VPN)

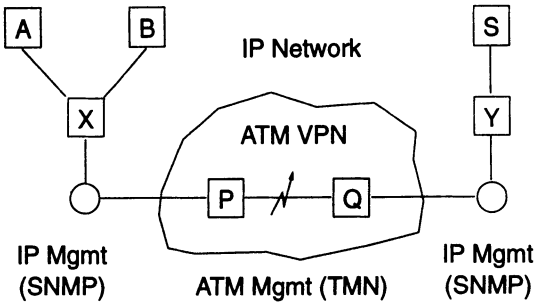

Figure 4 IP over ATM Example Scenario.

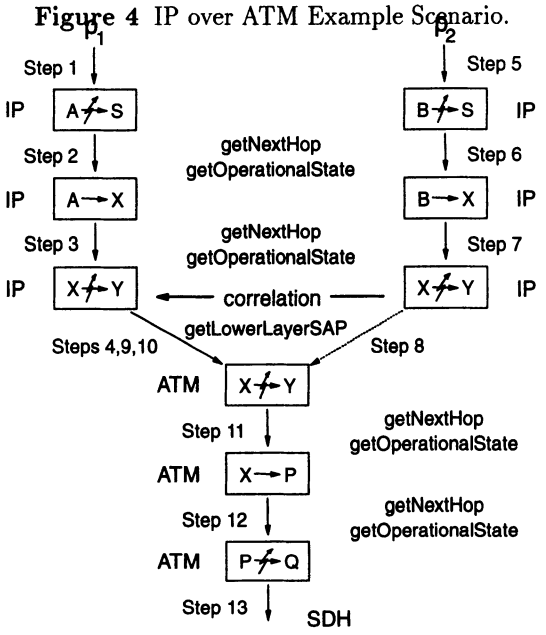

Figure 5 Steps Performed in the Example Scenario.

from a network provider. The ATM VPN part is managed by the network provider using the TMN management framework and an M.3100 based network model for the ATM network. Each domain is controlled by a separate intelligent agent that is capable of executing the fault isolation and event correlation algorithm independently.

The steps which the algorithm performs are shown in Figure 5. Assume a physical link failure in the ATM VPN between switches P and Q. As a consequence, no communication is possible from clients A and B with the server S. This problem is reported by the client A to the IP manager (Step 1). The network layer fault isolation creates a problem p_1 and starts the search algorithm for this problem in a separate thread. After exploring that the connectivity between A and X is

operational (Step 2), the thread detects the breakdown of the connection between the IP routers X and Y as the cause for the TCP connection problem (Step 3). The root cause of the current problem is updated to reflect this fact (Step 4) and the search will be continued because this problem has not yet been identified by another search thread.

Client B reports a communication problem on the TCP connection between B and S to the SNMP management as well. A new problem p_2 is created and a thread starts to diagnose the TCP connection (Step 5). This thread executes in parallel to the thread that is working on the TCP connection problem p_1 between A and S. The thread working on problem p_2 determines the cause for problem p_2 in the IP connectivity problem between X and Y (Steps 6,7). Since there is already an open problem registered for the managed object representing this connection in the agent, the thread appends (correlates) problem p_2 with problem p_1 and terminates (Step 8).

The thread for problem p_1 now migrates the problem to the intelligent agent responsible for the ATM VPN because the IP connection between X and Y is served by a path through the ATM WAN (Step 9; application of `getLowerLayerSAP`). It displays to the operator that a problem in the link between X and Y was detected and reported to the network operator running the link. A new thread is created in the ATM agent (Step 10) and the thread in the IP agent terminates or waits for a response from the ATM agent. This thread determines the `trail` object representing the ATM path that serves the IP connection between X and Y, checks its operational state, and all its server connections. It detects that there is connectivity between X and P (Step 11) but not between the switches P and Q (Step 12). This is the root cause of the problem and is notified to the fault application of all management domains that have dealt with symptoms of the problem (in this case, the fault application for the ATM VPN management domain and the fault application for the IP management domain at the client site). If the underlying SDH network has a management domain as well, the problem could be passed to this system and the search is continued there (Step 13).

6 CONCLUSION

A solution for fault isolation and event correlation in integrated networks is presented. It is based on the model traversing approach, i.e. it reconstructs fault propagation during run-time by investigating relationships between managed objects. The proposed algorithm has specific advantages compared to other fault isolation and event correlation solutions.

The algorithm is simple, i.e. its dynamic behavior is straightforward and can be reconstructed. Since a single symptom event is sufficient to trigger the fault localization procedure, the algorithm is robust against loss of events and is able to perform its task based on incomplete data. The algorithm can operate at very high speed because the kernel algorithm can run very efficiently in a distributed and parallel fashion. Experience has shown that the system is able to cope with event

storms of several hundreds of events per second. The technique was designed to incorporate other correlation techniques. This flexibility is of utmost importance for real-life environments with its integration of different networks with different requirements. The algorithm is generic, i.e. it is applicable for different network types and management architectures. The simple, well defined interface between the kernel algorithm and the Information Access modules easily supports different management paradigms and network protocols. The system has been implemented by IBM for SNMP-managed IP networks using MIB-II as a prototype and is being realized for TMN based systems using the M.3100 information model in a production level system. The experiences made during operation validated the approach.

Further studies on the applicability of the technique to other integrated networks are currently ongoing. Particular focus is put on the integration with expert systems and on dependency-graph based techniques for fault isolation on the application layer.

ACKNOWLEDGEMENTS

The authors would like to thank Prof. Kurt Geihs from the University of Frankfurt and the members of the IBM ENC Systems and Network Management Department for their helpful suggestions.

REFERENCES

Aidarus, S., Plevyak, T. (1994) *Telecommunications Network Management into the 21st Century.* IEEE Press, New York.

Bouloutas, A., Calo, S.B., Finkel, A. (1992) Alarm Correlation and Fault Identification in Communication Networks. *IBM Technical Report*, TR-17967.

Dreo, G. and Valta, R. (1995) Using Master Tickets as a Storage for Problem Solving Expertise. *Proc. 4th IFIP/IEEE International Symposium on Integrated Network Management*, 328-40.

Dupuy, F., Nilson, C., and Inoue,Y. (1995) The TINA Consortium: towards networking telecommunications information services. *IEEE Communication Magazine*, **33**, No. 11, 78-83.

Houck, K., Calo, S.B., Finkel, A. (1995) Towards a Practical Alarm Correlation System. *Proc. 4th IFIP/IEEE International Symposium on Integrated Network Management*, 519-30.

ITU (1995) Recommendation M.3100 Generic Network Information Model.

Jakobson, G. and Weissmann, M.D. (1993) Alarm Correlation, *IEEE Network*

Jordaan, J.F. and Paterok, M. (1993) Event Correlation in Heterogeneous Networks Using the OSI Management Framework. *Proc. ISINM'93, IFIP TC6/WG 6.6 International Symposium on Integrated Network Management*, 683-95.

Kätker, S. and Paterok, M. (1996) *System zur Überprüfung eines Datenübertragungsnetzwerks.* German Patent No DE 44 28 132 C 2, US Patent No GE 994 021 N.

Kätker, S. (1996) A Modelling Framework for Integrated Distributed Systems Fault Management. *Proc. IFIP/IEEE International Conference on Distributed Platforms*, 186-98.

Katzela, I. and Calo, S.B. (1995) Centralized vs. Distributed Fault Localization. *Proc. 4th IFIP/IEEE International Symposium on Integrated Network Management*, 251-61.

Kehl, W. and Hopfmüller, H. (1993) Model-Based Reasoning for the Management of Telecommunication Networks. *Proc. ICC'93, IEEE International Conference on Communications*, 13-17.

Kliger, S., Yemini, S., Yemini, Y., Ohsie, D., Stolfo, S. (1995) A Coding Approach to Event Correlation. *Proc. 4th IFIP/IEEE International Symposium on Integrated Network Management*, 266-77.

Lewis, L. (1993) A Case-Based Reasoning Approach to the Resolution of Faults in Communication Networks. *Proc. ISINM'93, IFIP TC6/WG 6.6 International Symposium on Integrated Network Management*, 671-82.

Nygate, Y.A. and Sterling, L. (1993) ASPEN – Designing Complex Knowledge Based Systems. *Proc. 10th Israeli Symposium on Artificial Intelligence Computing Vision, and Neural Networks*, 51-60.

Nygate, Y.A. (1995) Event Correlation using Rule and Object Based Techniques. *Proc. 4th IFIP/IEEE International Symposium on Integrated Network Management*, 279-89.

OMG (1995) The Common Object Request Broker: Architecture and Specification, Rev. 2.0.

Richter, L., Wakano, M., and Oshigiri, H. (1995) *Telecommunications Information Networking Architecture - Network Resource Information Model.* TINA-Consortium Document No TB_C2.LSR.001_1.0_93.

Stallings, W. (1993) *SNMP, SNMPv2, and CMIP; The Practical Guide to Network Management Standards.* Addison Wesley.

BIOGRAPHY

Stefan Kätker received his Diploma in Computer Science from the University of Erlangen-Nürnberg, Germany, in 1992. Since 1992 he is working in the Systems- and Network Management Department at the IBM European Networking Center in Heidelberg. Currently he is architect and scientific consultant for TMN fault management products and solutions. Research interests include SNMP- and TMN-based fault, application, and distributed systems management.

Martin Paterok received his Diploma and Dr.-Ing. in Computer Science from the University of Erlangen-Nürnberg, in 1985 resp. 1990. After working at IBM's T.J. Watson Research Center in Yorktown Heights he joined the IBM European Networking Center Heidelberg in 1991. He was staff member and manager of the Systems and Network Management Department. He is now manager of the Department for Mobile Data Communication. Research interests include integrated systems management and wireless data communication, in particular telematics and intelligent highway.