TECHNICAL RESEARCH REPORT

An Automated, Distributed, Intelligent Fault Management System for Communication Networks

by Hongjun Li, John S. Baras, and George Mykoniatis

CSHCN T.R. 99-30
(ISR T.R. 99-57)

The Center for Satellite and Hybrid Communication Networks is a NASA-sponsored Commercial Space Center also supported by the Department of Defense (DOD), industry, the State of Maryland, the University of Maryland and the Institute for Systems Research. This document is a technical report in the CSHCN series originating at the University of Maryland.

Web site http://www.isr.umd.edu/CSHCN/

AN AUTOMATED, DISTRIBUTED, INTELLIGENT FAULT MANAGEMENT SYSTEM FOR COMMUNICATION NETWORKS [1]

Hongjun Li, John S. Baras, and George Mykoniatis

Center for Satellite and Hybrid Communication Networks
Electrical Engineering Department and Institute for Systems Research
University of Maryland, College Park, MD 20742
E-mail: {hjli, baras, mykoniat}@isr.umd.edu

ABSTRACT

In this paper we present a Distributed Intelligent Fault Management (DIFM) system for communication networks. The overall architecture of the proposed system is based on a distributed cooperative multi-agent paradigm, with probabilistic networks as the framework for knowledge representation and evidence inferencing. We adopt the management by delegation paradigm [5] for network monitoring and integrate both hard and soft faults.

INTRODUCTION

To meet the needs of current and future communication networks, it is the responsibility of network management to maintain the network operations and service. The role of fault management is to detect, isolate, diagnose and correct the possible faults during network operations.

In current and future communication networks, which are broadband, very large, heterogeneous and complex, the dynamics become increasingly difficult to understand and control. In the cases of multiple faults, for example, it is almost impossible for the network manager, inundated in the ocean of alarms, to correlate the alarms and localize the faults rapidly and correctly just by his experience. Further, more and more users, possibly with different or even competing requirements of quality of service (QoS), wish to benefit from the networks. These will pose significant problems on fault management and thus an *automated* system with more advanced techniques is needed.

Knowledge-based expert systems have been very appealing for automated complex system fault diagnosis [2][7]. Nevertheless, most of the developed expert systems were built in an *ad-hoc* and unstructured manner by simply transferring the human expert knowledge to an automated system. Usually, such systems are based on deterministic models and they are designed to replace the human experts. Observing that the cause-and-effect relationship between symptoms and possible causes is inherently nondeterministic, *probabilistic* models [13] can be considered to gain a more accurate representation. Instead of replacing the human expert, the expert system based on such a probabilistic model is expected to behave as the *assistant* to a human expert by providing processed information and suggestions timely and automatically.

In a centralized manager-agent paradigm, all of the symptom information has to be sent to the central manager for processing. Such a paradigm works well for small networks. But as the networks become larger, it will result in vast amounts of communication between manager and agent and thus occupy too much bandwidth unwisely. Since there are many cases where the faults can be resolved *on-the-spot*, we propose that the faults should be handled *locally* if they are local. Only those that cannot be handled locally should draw global attention. Observing also that communication networks are hierarchical and distributed by nature, it is most desirable to take a multi-layered architecture and distribute some intelligence to the lower layers that are closer to the managed objects. The entities, which have the distributed intelligence and whose responsibilities are fault diagnosis in the local domains, are referred to as "Intelligent Agents"(IA) [10].

In previous research, the term "fault" was usually taken the same as "failure", which means component

[1] Prepared through collaborative participation in the Advanced Telecommunications/Information Distribution Research Program (ATIRP) Consortium sponsored by the U.S. Army Research Laboratory under the Federal Research Laboratory Program, Cooperative Agreement DAAL01-96-0002. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon. Research also supported by NASA under NASA Cooperative Agreement NCC3-528, and by a grant from Lockheed Martin Telecommunications.

malfunctions, e.g. sensor failures, broken links or software malfunctions. Such faults are called "*hard* " faults and can be solved by replacing hardware elements or software debugging and/or re-initialization. The diagnosis of the "hard" faults is called *"re-active"* diagnosis in the sense that it consists of basically reactions to the actual failures. In communication networks, however, there are still some other important kinds of faults that need to be considered, for example switch performance degradation and link congestion. Since there might not be a failure in any of the components, we call such faults "*soft* " faults. "Soft" faults are in many cases indications of serious problems and for this reason, the diagnosis of such faults is called *"pro-active"*diagnosis. By early attention and diagnosis, such pro-active management will sense and prevent disastrous failures. In our system, we consider both hard and soft faults, with emphasis on the latter.

SYSTEM ARCHITECTURE

Figure 1 shows the general architecture of our DIFM system. The managed network is divided into several domains and for each domain, there is an intelligent agent attached to it. A domain is an abstract notion, which might be a subnet, a cluster, a host or a member of a functional partition. For those problems that none of the individual agents can solve, there is a mechanism by which the agents can report to the coordinator and share the information in order to get a global view and solve it cooperatively. So the whole system is, from the agent point of view, a distributed, cooperative multi-agent system.

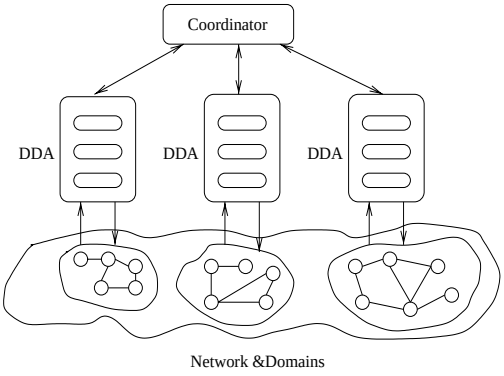

Figure 1. Architecture of DIFM

Each agent is called a "Domain Diagnostic Agent (DDA)" with the goals of monitoring the health of the domain, and diagnosing the faults in a cost-efficient manner. Each local DDA in DIFM includes the following three functional components: Fault Detection and Classification (FDC), Fault Localization and Identification (FLI), and Fault Corrections (FC), respectively.

Fault Detection and Classification: The inputs are measured data, alarms or users reports. Such inputs are analyzed so that the current system behavior is obtained, based on which the fault hypotheses can be generated and tested. The model of "normal" behavior will be stored explicitly or implicitly, and model parameters should be adapted (learned) along the way. The output of the FDC is the type of fault(s), for example call failure.

Fault Localization and Identification: The principal operation of FLI is to determine what might be the primary *causes* for the symptoms (fault types) recognized by the FDC. Since the relationship between symptoms and causes is probabilistic in nature, *probabilistic* fault models should be considered. Belief networks, which form the basis for probabilistic reasoning and expert systems, manifest themselves as the most suitable choice [13]. Other examples of using probabilistic models for fault diagnosis can be found in [4] and [6].

Fault Corrections: Given the possible causes, we are going to generate a set of tests or repair sequences based on some heuristic or decision-theoretic strategies. This is basically a *sequential decision process* and thus can be formulated mathematically in some careful way as a Markov decision problem.

The above functional components are implemented by the following physical components: Intelligent Monitoring and Detection Assistant (IMDA), Intelligent Domain Trouble-shooting Assistant (IDTA), and Intelligent Communication Assistant (ICA). Many such DDAs will then be distributed in the network and act as the "local experts" for different domains.

SYSTEM COMPONENTS

Intelligent Monitoring and Detection

IMDA is in the lowest layer and is the interface to the Network Elements (NE). It provides symptoms information to IDTA, as illustrated in Figure 2. FDC is implemented here.

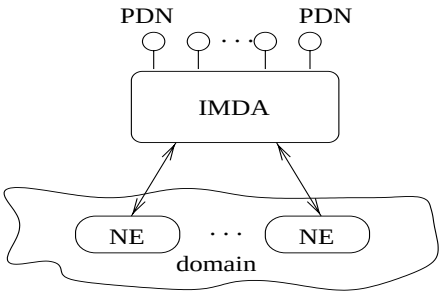

Figure 2. Illustration of an IMDA

IMDA monitors the network elements that belong to its network domain. Based on this information it identifies the specific type of the fault that has occurred in that network domain. Specifically, for each IMDA we define a number of outputs and relevant activation status, each one modeling a certain type of fault. We call them Problem Definition Nodes (PDNs). We define five activation levels for each PDN to reflect the severity of the relevant network fault. Those five severity levels are "alarm", "major", "minor", "warning" and "normal". Note that the set of PDNs should be defined carefully to reflect the most typical kinds of problems and the mapping to the monitored information is actually a pattern recognition problem. The definition of the set of PDNs depends on the specific network technology (eg. ATM, IP).

From the above description it becomes obvious that there is an explicit need for network monitoring. IMDA requires for its operation a good knowledge of the current status of the various network elements that comprise its network domain, in order to be able to set accurately the status of the various supported PDNs. Since we are mostly interested in soft faults, the network monitoring system has to support not only detection of network failures but also detection of network performance deterioration. We believe that the classical polling-based model for network monitoring is not adequate because it will introduce a large amount of management traffic, especially for high speed networks. So IMDA will be based on the management by delegation paradigm [5].

Specifically, we define a number of appropriate network monitoring policies for supporting the monitoring needs of IMDA. Those monitoring policies can be realized using one of the existing technologies like Java, CORBA, or even OSI management [1][12]. Each monitoring policy is introducing a number of monitoring *sensor-objects* inside the management components of the various network elements (or at least as close as possible). Those sensors are actually place-holders for monitoring intelligence. They can be responsible for watching a specific number of raw information objects or even other monitoring sensors, and when some appropriately pre-defined condition becomes valid they are able to notify IMDA. A classic example of such a condition is a threshold crossing of a monitored network statistic. Its value can be computed from raw statistical information stored in the NEs.

Using such a network monitoring paradigm we can realize the operation of IMDA directly in the network management application running on each monitored network element in the network domain. The notifications that will be emitted by the created sensor-objects will directly be mapped to the supported PDNs. Avoiding the classical polling-based network monitoring results in reduced network monitoring traffic and allows the management entities to focus in the realization of the specific monitoring policies needed by IMDA for supporting the PDNs.

Intelligent Domain Trouble-shooting

The IDTA is located above IMDA and acts as the trouble-shooter for the symptoms reported from the IMDA. FLI and FC are implemented here. It includes a probabilistic expert system, which is basically a belief network database. Based on the activation status of the PDNs, a sub-belief network is extracted from the database and then the inference and trouble-shooting begin, as described below and shown in Figure 3. The inputs are the activation status of the PDNs and the outputs are primary causes and the suggested test sequence. The IDTA functionality includes scheduling of the PDNs, extraction of the sub-belief network, inference and trouble-shooting. They consist of a trouble-shooting cycle.

At the same time, there might be more than one PDNs that are not in the "normal" state. The "alarms" are to be considered with highest priority and the "warnings" with the lowest (the "normal" status incurs no diagnosis at all). We notice that there should be a mechanism to discriminate the severity levels and determine for which PDNs the sub-belief network will

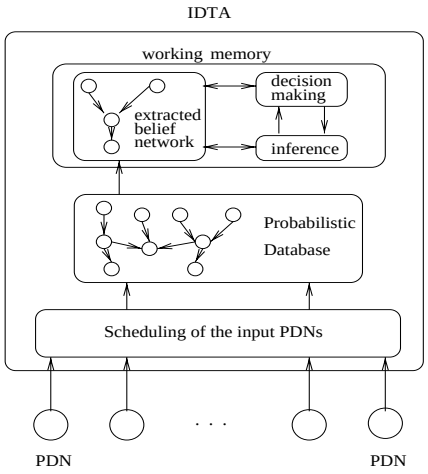

Figure 3. Illustration of an IDTA

be extracted. For example, in a case where PDN one is in "alarm" status and PDN two is in "minor" status, it might be more desirable to take care of PDN one only instead of considering both of them (let alone PDN two only). The scheduling algorithm is for further study.

For the selected PDNs, a sub-belief network can be extracted into the working memory. This can be done using the idea of *d-separation*, as defined in the introduction to belief networks in [13][14]. The nodes extracted are those that are not d-independent of the selected PDNs. In other words, they are related in some way to the problematic PDNs and hence, the candidates as the causes of those symptoms.

Given the extracted belief network, the belief of any non-PDN node to be faulty can be calculated through backward inference, based on which static or dynamic trouble-shooting strategies can be adopted to generate the test sequence. To do this, it is desirable that a suggestive test sequence be proposed by the system based on some decision-theoretic principles. The principle we adopt is to minimize the expected value of some appropriately defined cost function which takes into consideration labor, time and urgency factors. Such a problem is basically a Markov decision problem and in [3] we provide a dynamic programming formulation and the proposed solution schemes. Reinforcement learning techniques are also investigated there.

Re-actions are embodied in the handling of the alarms. For pro-actions, however, we note the following: First, since the "abnormal" PDNs with status other than "alarm" can also be dealt with, the diagnosis afterwards is actually **pro-diagnosis** in the sense that it is dealing with something before it really goes wrong, assuming proper definition of the PDNs; Second, since the belief network nodes are not restricted to be physical entities, they can also be "logical" or performance nodes, such as "link congestion", so that "soft" faults can also be included.

Intelligent Communication

When the problems cannot be solved by any of the individual DDAs, it is the role of the ICA to report the problems to an upper layer, where correlation and coordination can be done and a conclusion can be drawn from a global point of view. The ICA is illustrated in Figure 4.

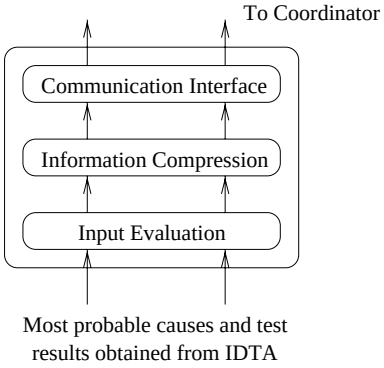

Figure 4. Illustration of an ICA

The inputs are results of belief computations (the most probable causes) for various extracted belief networks and results from test sequences. The outputs are compressed versions of symptom statistics and of the results given as inputs. The outputs are then transmitted to a coordinator in the upper layer via some communication links.

The ICA functionality includes assessment of the value of the results from belief computations and test sequences (Input Evaluation). This evaluation will decide to what extent it is worthwhile to send these results to an upper layer. Only the most relevant results will be sent. In addition, the ICA will have a function to select features and compress the data describing valuable inputs (Information Compression). The evaluation and compression will help reduce the amount of data to be transmitted and thus reduce the bandwidth

overhead for such communications. Finally, the ICA must include a function which will decide where to send the compressed descriptions and how to communicate with minimum overhead with the upper layer (Communication Interface). To understand such selected and compressed information (encoded data), the coordinator receiving such information must share with ICA the same encoding-decoding protocol. The whole communication will be realized using appropriate CORBA or Java based technology [11].

CONCLUSIONS [2]

In this paper we presented a Distributed Intelligent Fault Management (DIFM) system for communication networks. For more details, we refer to [3][9]. Note that in a communication networks environment, the response time of DIFM has to be ultra-short in order to manage the system meaningfully. So most of the system functions here are supposed to be well-trained *off-line* before they are applied. The off-line training usually includes identification of the PDNs, determination of the belief network structure and estimation of the belief network parameters, etc. The *on-line* learning capability serves as the complementary characteristic when prompt adaptation is needed. But the reliance on this should be limited. Observing also that the system architecture is hierarchical by nature and the notion of domain is quite generic, this system design can be naturally applied to a multi-layered, hierarchical practical communication network (such as IP over ATM), with the notion of domain appropriately identified.

References

[1] N. Anerousis, A. Biliris, "An Architecture for Creating Scalable, Web-based Management Services", AT & T Labs Research, 1998

[2] J. S. Baras, M. Ball, S. Gupta, P. Viswanathan and P. Shah, "Automated Network Fault Management", *MILCOM'97* , Monterey, CA, November 2-5, 1997

[3] J. S. Baras, H. Li and G. Mykoniatis, "Integrated, Distributed Fault Management for Communication Networks", Technical Report, CSHCN TR 98-10, University of Maryland, 1998

[4] R. H. Deng, A. A. Lazar, W. Wang, "A Probabilistic Approach to Fault Diagnosis in Linear Lightwave Networks", *IEEE JSAC,* vol. 11, no. 9, pp. 1438-1448, 1993

[5] G. Goldszmidt, Y. Yemini, "Distributed Management by Delegation", *Proc. of the 15th Inter. Conf. on Distributed Computing Systems,* 1995

[6] C. Hood and C. Ji, "Pro-active Network Fault Detection", *Proc. IEEE INFOCOM,* 1997.

[7] L. Kerschberg, R. Baum, A. Waisanen, I. Huang and J. Yoon, "Managing Faults in Telecommunications Networks: A Taxonomy to Knowledge-Based Approaches", IEEE, pp. 779-784, 1991

[8] H. Li, and J. S. Baras, "An Introduction to Belief Networks", CSHCN Technical Report, University of Maryland, 1998

[9] H. Li, G. Mykoniatis, and J. S. Baras, "Distributed Intelligent Fault Management for Communication Networks", submitted to IM'99, 1998

[10] T. Magedanz, K. Rothermel and S. Krause, "Intelligent Agents: An Emerging Tech-nology for Next Generation Telecommunications?", Proceedings of the 1996 IN-FOCOM, San Francisco, CA, 1996.

[11] Object Management Group, "The Common Object Request Broker Architecture and Specification", Rev. 1.2, Dec. 1993.

[12] G. Pavlou, G. Mykoniatis and J. Sanchez-P, "Distributed Intelligent Monitoring and Reporting Facilities", *IEE Distributed Systems Engr. Journal,* Special Issue on Services for Managing Distributed Systems, 1996

[13] J. Pearl, *Probabilistic Reasoning In Intelligent Systems: Networks of Plausible Inference,* Morgan Kaufmann, 1988

[14] S. Russell and P. Norvig, *Artificial Intelligence - A Modern Approach,* Prentice-Hall, 1995

[2]The view and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government.