

An Alarm Management Approach in the Management of Multi-Layered Networks

Jane Zupan and Deep Medhi
Computer Networking Research Lab
University of Missouri–Kansas City
Kansas City, MO 64110 USA
Email: jzupan@free.fr, dmedhi@umkc.edu

Abstract—We consider large interconnected networks that operate under a user-provider paradigm (such as IP over SONET) where networks from different layers are in different administrative domains. In such a relationship, the overall network survivability can benefit from a limited sharing of network management data between networks of different layers.

This work proposes a multi-layered alarm management framework in such a setting. The Alarm Manager uses a finite state machine to represent an alarm instance with state transitions triggered as a result of correlation of multiple alarms. We present a constructive argument to demonstrate the correctness of the finite state machine. Alarm correlation is handled by a rule-based reasoning engine. The set of correlation rules and corresponding actions depends on the definition of alarms that can enter the system. Therefore, three alarm categories are created based on the multi-layered paradigm and the inclusion of a performance management agent: provider network alarms, user network alarms, and predictive alarms.¹

I. INTRODUCTION AND MOTIVATION

As computer networks continue to grow in size and complexity, there arises a corresponding need to extend network management capabilities. Large networks are typically hierarchically structured in multiple layers, such as IP over SONET, IP over ATM over SONET, or IP over DWDM. Currently existing network management systems operate often only on one administrative network layer. In many cases, this is a system architectural decision. By contrast, a network management system that operates over multiple network layers, with an exchange of limited information from each network, can provide services to increase the efficiency of the overall network. Such services could include aggregation and correlation of fault and traffic monitoring data to result in more efficient, and thus faster, fault management, reconfiguration management, and performance management. In addition, many fault management tasks can be automated to a greater extent than possible with a single-layer network management system. This work examines the specific issues surrounding alarm management in a multi-layer network management paradigm and proposes a solution in the design of an Alarm Manager to operate in a multi-layered network management system.

¹Supported by DARPA and Air Force Research Lab, agreement No. F30602-97-1-0257.

A. Multi-Layered Network Management

We view a network as an independently operated administrative domain that provides a network service. In this context, there are service providers and service users, which need not be geographically or administratively connected. Furthermore, a network can serve in both capacities: providing a service to one network, and using the services of another network. For example, in a network running IP over ATM over SONET, ATM is both a service provider and a service user. For clarity, throughout this work, we will denote the network service provider as a *Provider* Network and a network service user as a *User* Network.

As a concrete example, consider an Internet Service Provider (ISP). An individual ISP often operates its own switches/routers and links. Typically, it will be connected to an underlying network, e.g. SONET, for physical transport. In this case, the SONET network is the provider network and the ISP is the user network. A failure in the SONET network can affect the ISP's operation. However, if the administration of the network is independent of the ISP, there is no existing structure for coordination of such a fault. There may be independent network management systems on the individual network layers, but in general, there is not currently an established system for coordinating network management functionality across network layers.

The purpose of a multi-layered network management architecture is to provide a loosely coupled framework for addressing inter-layer issues in the context of network survivability. Such a cooperative framework can be used to increase the efficiency of fault management and performance management for the *overall* network, and has been presented in the context of survivability of multi-layered networks in [5].

Figure 1 depicts the distributed, hierarchical design of the multi-layered network management framework. Each administrative domain has one or more network Agents that obtain management data from the nodes and report to the domain Manager. The primary task of the Manager is to aggregate management data and report to the next level in the hierarchy, the Across Layer Manager of Managers (ALMoM). As with the Agents and Managers, there can be more than one ALMoM to provide for scalability of the architecture. The Managers and Agents each have two interfaces, defined functionally:

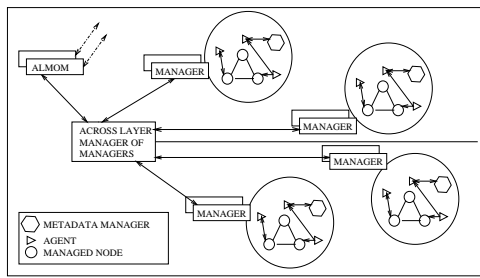


Fig. 1. A Multiple-Layer, Distributed Hierarchical Network Management Architecture

a *horizontal interface* obtains data from the domain-specific Agents, and a *vertical interface* reports data to the next level in the hierarchy. The ALMoM receives network management information from the Managers of multiple domains and provides services for the managers. The primary role of the ALMoM is to use the available data to coordinate recovery mechanisms between layers that would otherwise individually take redundant and sometimes unnecessary actions.

B. Literature Review

The following six techniques are commonly used to model event correlation by commercially available event/alarm correlation systems [4], [6]: (1) rule-based reasoning, (2) model-based reasoning, (3) case-based reasoning, (4) codebook, (5) state transition graph model, and (6) finite state machine model. It should be noted that none of these six models explicitly incorporates multiple-layer, or spatial, information. To our knowledge, commercial applications that use these six techniques for event correlation operate often on just a single network layer.

In [2], Jakobson and Weissman considered alarm correlation to be a "relatively new" process of real-time network management. Their model is interesting in its hierarchical nature and functional divisions. Bouloutas, Calo, and Finkel, [1], associate explicit fault-localization information with each alarm. The purpose of using a model that allots a certain amount of intelligence to an alarm is to provide scalability and extensibility to the alarm correlation process.

Jakobson, et al., [3] state that event correlation should include different functional domains, such as alarm/fault, performance, security, and configuration functions, referred to as *cross-correlation*. The idea of *proactive*, rather than *reactive*, anomalous event detection is discussed in [7]. Such proactive fault detection in a network can be used by a network management system to detect and provide automated resolution for faults before they actually occur. The agents used to perform the proactive alarms are addressing the functional network management category of performance management. The agent in this study uses statistical methods to detect anomalies and the Alarm Manager incorporates both fault and performance management information.

In summary, the above systems and models all have certain strengths. Our work proposes a model for an Alarm Manager,

designed to operate with the previously developed multiple-layer network management architecture [5]. The Alarm Manager is unique in that it addresses both the temporal and the spatial aspect of alarm management.

II. DESIGN OF A MULTI-LAYERED ALARM MANAGEMENT SYSTEM

In a multi-layered network management environment, alarm management coordinates traffic provisioning and network recovery due to failure. With the availability of limited information from multiple networks in the stack, the goal of alarm management is to maximize efficiency of recovery mechanisms for all layers in the network and to provide graceful degradation when recovery is not possible. This work proposes an Alarm Manager that loosely incorporates the functional network management categories of fault management, performance management, and configuration management in the context of a multi-layer network management architecture. The primary areas addressed in the design of the Alarm Manager are:

- 1) The modelling of a single alarm,
- 2) Communication between multiple alarms, and
- 3) The method of correlation of the alarms.

The representation of a single alarm is the central component of the Alarm Manager. Because one alarm can trigger other alarms, some form of communication between the alarms must be included in the model. The most effective representation for an alarm is therefore simple, concise, and incorporates inter-alarm communication. This work proposes a finite state machine model to represent each alarm and to define the communication between alarms. We provide a constructive argument to confirm the correctness of the finite state machine and thus the model for the alarm and operational validity of the system.

Correlation of alarms is another key component to the correct operation of the Alarm Manager. Correlation in a multiple-layer environment, with the inclusion of predictive traffic information, introduces new issues that are not present in single-layer alarm management systems. The Alarm Manager categorizes alarms from three types of sources: a Provider Network Agent, a User Network Agent, and a Predictive Agent, which triggers alarms based on traffic analysis from the physical network. We use a rule-based reasoning model to correlate the alarms from these three sources and define the corresponding set of rules. The actual correlation is triggered as a result of specifically defined combinations of events involving multiple alarm finite state machines.

A. Definition of Alarms

The Alarm Manager examines a subset network-specific events, but from a multiple-domain perspective. Furthermore, this subset of events is reduced to only those network alarms whose resolution performance can be improved, for the overall network, with across-domain communication. A key assumption precedes the development of the Alarm Manager finite state machine: the alarms are filtered at the Network Agent

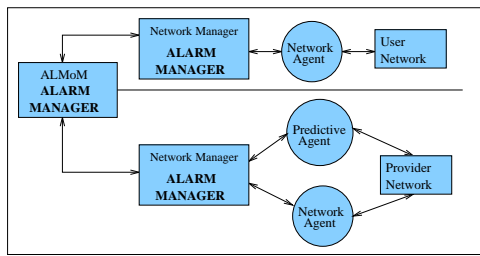


Fig. 2. The Alarm Manager in the Multiple-Layer Network Management Architecture

level. Therefore, when they are received by the Alarm Manager, the origin and type of the alarm is known. Alarms come from three sources: a Provider Network (Provider Network Alarm), which is assumed to be a physical-layer network, a User Network (User Network Alarm), which is assumed to be a logical-layer network, or a Predictive Agent (Predictive Alarm), which collects information about physical layer links.

B. Integration Into the Multi-Layered Network Management Architecture

Figure 2 shows that the Alarm Manager sits on the management level of the multiple-layer network management architecture. A predictive bandwidth provisioning agent on the physical layer of the network addresses performance management functions within the overall network system.

The purpose of the Alarm Manager is to facilitate the fault management and performance management for multi-layered networks. With those functions as a guide, the communication paths of an alarm can be traced in figure 2. From a fault management perspective, the Network Agent (from either the User or Provider Network) perceives a fault either through polling or a trap. This alarm is forwarded to the Network Manager. The alarm is categorized to fit into the structure of the Alarm Manager, and forwarded to the Alarm Manager in the ALMoM. The specific processing of the Alarm Manager, at both the ALMoM and Network Manager levels, is described in detail later.

The Predictive Agent provides performance management functions. Its role involves proactive bandwidth provisioning, according to dynamic traffic characteristics. From a high-level view of the Alarm Manager, the predictive alarm follows the same path as the Network Agent alarm. The additional information provided to the Alarm Manager from the predictive alarms is correlated with the fault-related alarms to give a comprehensive view of the recovery mechanisms of the overall network. With this aggregated data, the Alarm Manager can generate communication between layers and eliminate some unnecessary alarm resolution, thereby increasing efficiency of the overall network.

C. Specifications for the Alarm Manager

The services to be provided by the Alarm Manager are divided into two categories: primary specifications must be accommodated, and secondary specifications are not mandatory

in the first version, but may serve as a guide for developing the next version of the Alarm Manager. Primary specifications are:

- 1) Receive an alarm from an agent,
- 2) Classify the alarm,
- 3) Correlate alarms across layers and temporally,
- 4) Perform recovery procedures when applicable,
- 5) Perform smoothing or hold functions when applicable, and
- 6) Perform reconfiguration when applicable.

Secondary specifications are:

- 1) Incorporate a learning capability so that the correlation step can recognize previous patterns and perform the next step based on experience, and
- 2) Add a further degree of alarm filtering at the network manager level.

The first two specifications, receiving and classifying an alarm, translate into initial operations that prepare the alarm for further processing. The other four primary specifications are each represented as a state in the finite state machine.

D. Alarm Sources

The Alarm Manager receives alarms from three primary sources: the Predictive Agent, the User Network Agent, and the Provider Network Agent. These sources all produce different types of alarms that are potentially related to each other. Correlating these alarms temporally and across layers could decrease the need for recovery mechanisms, thereby increasing the efficiency of the network as a whole. For example, in an IP/SONET network, if a link goes down on the SONET layer, the IP nodes may perceive a corresponding logical link failure and trigger a recovery process. Since the network management system maintains some amount of knowledge of both networks, the relationship between alarms from both layers can be determined. Instead of both layers triggering recovery mechanisms, only the SONET layer can perform fault recovery, thereby saving the overhead of the IP layer message-passing to find an alternate route.

The Alarm Manager handles a finite set of alarms from each of the three source agents. Although the three sources of alarms all generate different types of alarms, the actions triggered by the alarms can be abstracted and coordinated across layers in order to minimize unnecessary alarm resolution. The finite state machine represents this process. Before presenting the Alarm Manager finite state machine, the nature of the three types of alarms will be discussed.

1) *Provider Network Alarms:* In this network management system, provider networks are the lower layer networks that provide the physical transport for the upper layer networks. For example, in an IP over SONET environment, the SONET layer is the provider network and the IP layer is the user network. In a provider network environment, alarms typically involve the functionality of a link. Although the fault may be in an interface, a link, or elsewhere, from the point of view of the traffic, the alarm refers to a specific link being affected.

From the perspective of the Alarm Manager, Provider Network Alarms are related to a specific link.

2) *User Network Alarms*: User networks are the upper layer networks that, when viewed as a solitary entity, are connected by logical links. That is, the traffic does not physically flow on the user network, rather, it proceeds through the layered stack to the physical layer for actual transport. The user network, then, has its own types of alarms that are different from the alarms of a provider network.

The user network typically interprets alarms as affecting either logical links or physical nodes. An alarm related to the functionality of the node itself could be caused by overload of the CPU, or by the actual failure of the node.

If the user network is running a dynamic routing protocol, logical link failures are signaled with routing messages; for example, consider an IP network running OSPF protocol. Such a failure-signalling message would cause the protocol to initiate a series of messages in order to search for an alternate route between two nodes. Packets are dropped while the protocol resolves the logical link failure, and the measure of dropped packets could trigger an alarm. In a static routing environment, logical link failures also trigger alarms. Alternate routes will be found either manually, or through an automated process.

From the point of view of the Alarm Manager, the type of User Network Alarms that are addressed represent a problem with logical link connectivity. The problem will probably be related to a physical link fault and the Alarm Manager will determine this connection.

3) *Predictive Alarms*: The Predictive Agent, which analyzes traffic patterns, is located at the physical layer. Like Provider Network Alarms, alarms generated by the Predictive Agent refer to a specific link. The Predictive Module measures traffic, and provides a predictive measure of bandwidth needed every time period. For example, a typical time period could be 5 minutes. If no change in bandwidth is foreseen if the next time interval, there will not be an alarm generated by the predictive module. The amount of bandwidth needed by a traffic flow could represent either an increase or a decrease in the current amount of bandwidth provided.

Predictive Alarms can be related to Provider Network Alarms. If a Predictive Alarm is received for a link, then a Provider Network Alarm is received for the same link, resolution for the Predictive Alarm should be stopped and the Provider Network Alarm resolution will take priority. For this reason, alarm correlation must be performed when receiving a new alarm and the Alarm Manager must decide how to handle this information efficiently.

III. THE ALARM MANAGER FINITE STATE MACHINE

Using a finite state machine to model the Alarm Manager allows a fairly complex system of communication between components to be represented in a concise format. Sequences of events and multiple conditional events can be portrayed succinctly as a progression from one state to the next. The actions that trigger a state change show the communication

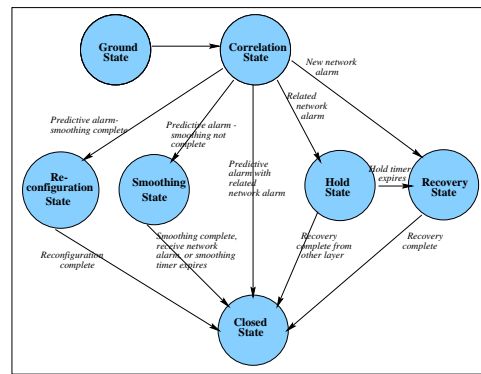


Fig. 3. The Alarm Manager Finite State Machine

between components. Such a modular representation of a complex system of communication lends itself smoothly to implementation. Furthermore, the individual states each have a well-defined functionality. Clearly specified interfaces between the states result in a modular system where the internal design of any state can be enhanced and extended without adversely affecting the operation of the alarm management system as a whole. This is not to suggest that building such a finite state machine from each network's knowledge feeding into the Alarm Manager is an easy exercise; such a construction is outside the scope of this paper.

The limitation of the finite state machine is that it must be carefully designed in order to avoid indefinite processing and communication deadlock. Two causes of such a scenario include: a loop in the design of the finite state machine, and the possibility that processing in a state does not terminate. Deadlock can be a problem in a finite machine if one process is caught waiting indefinitely for another process to finish.

The Alarm Manager operates on both a temporal plane and a spatial plane by receiving successive alarms from different network layers. The Alarm Manager model must capture both the spatial and temporal aspects of the alarms. Because the states of the finite state machine can be represented functionally, and the actions triggered by the states result from both temporal and spatial triggers, the finite state machine is able to capture both dimensions of the operations of the Alarm Manager.

Figure 3 shows the Alarm Manager Finite State Machine. For clarity, the only events that are presented are those that trigger a state transition. Each state also has a series of actions associated with it that are described in the following sections.

A. Description of the States

1) *Ground State*: This is the initial starting point of the finite state machine. When an alarm is received, an alarm instance is created in the ground state. The alarm is categorized in this state according to its source: Provider Network Alarm, User Network Alarm, or Predictive Alarm. Then the alarm is forwarded to the correlation state.

2) *Correlation State*: The correlation state must determine the relationship of a new alarm to other working alarms.

Related alarms are defined in this work from a layered network perspective. That is, if two alarms refer to the same physical link, then they are considered to be related. The actual correlation is a simple comparison of link IDs (i.e. source/destination address pairs). The Configuration Manager of the multi-layered network management architecture maintains a mapping of logical links to physical links, thus allowing the comparison of alarm link IDs across network layers.

The task of the correlation state is to determine the next state of an alarm with the knowledge of the states of other unresolved alarms in the overall network. For example, given that there is already a Provider Network Alarm in the system (i.e., it has not yet been resolved), and a User Network Alarm referring to the same physical link arrives in this system, the correlation state must decide the next state of the User Alarm. Additionally, the arrival of the new alarm may or may not trigger a state change for the original Provider Network Alarm. A set of rules, described subsequently, determines the association between the current state of the system and the next state of the incoming alarm.

3) *Recovery State*: If the incoming alarm is not related to another alarm, and it is due to a network fault, it is considered a *new* alarm and enters the recovery state. As it enters recovery state, two actions are triggered:

- 1) The hold timer is started,
- 2) Technology-dependent recovery procedures are begun.

The main function of the hold timer is to prevent more than one layer from initiating recovery mechanisms simultaneously when such an overlap in recovery is unnecessary. The hold timer is referenced by any subsequent alarms that are determined to be related to the initial alarm. If a relationship is found, the subsequent alarm will be placed in hold state, and will not start its own recovery procedures, unless the hold timer expires and recovery of the first alarm is not complete.

4) *Hold State*: The hold state is a temporary repository for Provider or User Network Alarms that have a related alarm, either from another layer or from an interface on the same link, in the recovery State. Alarms in this state do not undergo any processing; they only wait for an outside trigger to move them to the next state. The possible triggers include:

- 1) The hold timer expires, or
- 2) The alarm in recovery state completes its alarm resolution.

In the first case, the hold timer expires but the other related alarm is not resolved. If this happens, the current alarm should initiate its own recovery mechanisms, and therefore it moves to the recovery state. This is an unusual scenario. If the hold timer is set appropriately, this case should occur rarely.

In the second case, the other related alarm did fully complete its alarm resolution before expiration of the hold timer. Thus, the current alarm does not need to initiate recovery because its direct correlation to the first alarm means that the current alarm is resolved when the first alarm is resolved. This scenario illustrates the merit of this Alarm Manager. Since communication between layers exists, alarms occurring on

multiple layers can be resolved in a single layer. Unnecessary recovery processing can thus be avoided on one or more layers. Additionally, physical layer recovery typically occurs much faster than User Network layer recovery.

5) *Smoothing State*: The purpose of having a smoothing function is to avoid the potential overreaction to multiple Predictive Alarms. Smoothing in this system is defined as receiving three Predictive Alarms requesting bandwidth adjustment in the same direction (increase or decrease) *and* there are no related network alarms existing or received in that time. Predictive Alarms received at the correlation state before the smoothing is complete go to the smoothing state; the Predictive Alarm received that completes the smoothing advances to the reconfiguration state. If there is a related network alarm in the system, then the Predictive Alarm goes directly to the closed state, because reconfiguration of a link that is in the recovery process is unnecessary. Additionally, if a network alarm is received while there is a Predictive Alarm in the smoothing state, then the new alarm triggers the Predictive Alarm to advance to the closed state.

Like the hold state, the smoothing state maintains a timer. If a series of alarms does not complete the smoothing function and trigger reconfiguration within a set amount of time, the original alarm in smoothing state is closed and the smoothing process must start over with the next alarm. The timer guarantees that an alarm does not remain indefinitely in the smoothing state.

6) *Reconfiguration State*: The reconfiguration state is used uniquely by Predictive Alarms for bandwidth adjustment. If the correlation state determines that the smoothing procedure for a series of Predictive Alarms is complete, then the previous Predictive Alarms in that series progress from the smoothing state to the closed state, and the Predictive Alarm that caused the smoothing to finish moves to the reconfiguration state.

The action taken in this state is bandwidth adjustment of a flow. The Predictive Alarm determines the amount of bandwidth to be increased or decreased. The Predictive Agent knows the maximum allowable bandwidth for a traffic flow, and therefore will not generate a request for more than this maximum. After the bandwidth has been adjusted, the alarm proceeds to the closed state.

7) *Closed State*: This is the final state for an alarm in this system. The recovery state, hold state, smoothing state, correlation state, and reconfiguration state all send alarms to the closed state. Once an alarm has arrived at this state, the Alarm Manager has performed the level of alarm resolution that it is capable of for this particular alarm. When an alarm arrives here, it is removed from the alarm management system.

B. Correctness of the Finite State Machine

In order to verify that this finite state machine does not suffer from the pitfalls of looping and communication deadlock, and to show the correctness of the finite state machine as a protocol for the Alarm Manager, we present a constructive correctness argument of the finite state machine. The approach to this argument is two-fold: one, we show that there is no loop

possible between states, and two, that advancement out of each state is guaranteed.

Viewed as a graph, the source of the finite state machine is the ground state and the sink is the closed state. Consider the progression through the finite state machine as if a token started at the beginning and advanced through the graph. Refer to figure 3.

- 1) From the ground state, there is only one exit and no re-entry. Since the exit is triggered as an action of the ground state, and the actions in the ground state do not depend on other processes in other alarms, the token cannot remain in the ground state. The token thus advances directly to the correlation state.
- 2) Since the only entry into the correlation state is from the ground state, and there is no possible re-entry to either the ground state or the correlation state from any other state, no loop can be created at or before the token arrives in this state. The actions triggered at the correlation state do not depend on the processing of another alarm. Therefore, progression of the token out of the correlation state is guaranteed.
- 3) Since the only point of entry into the hold state is from the correlation state, there is no re-entry to the hold state and no loop can be created at or before the hold state. Alarms in the hold state depend on the processing of the alarm in recovery state. However, the hold timer, which is set when the other alarm enters the recovery state, limits the interdependence of the alarms. If the recovery actions in the recovery state are completed before the expiration of the hold timer, the token advances to the closed state. If the recovery actions are *not* completed before expiration of the hold timer, the token advances to the recovery state. Since the hold timer is finite, exit from the hold state is guaranteed.
- 4) Since the only exit from the recovery state is to the closed state, which is the sink of the graph and from which there is no possible re-entry into the graph, there can be no loop back into the recovery state. The technology-dependent recovery actions triggered in the recovery state must be finite and are time-limited. Therefore, exit from the recovery state is guaranteed.
- 5) Since the only exit from the reconfiguration state is to the closed state, there can be no loop back into the reconfiguration state. The actions triggered in the reconfiguration state must be finite and time-limited. Therefore, exit from the reconfiguration state is guaranteed.
- 6) Since the only exit from the smoothing state is to the closed state, there can be no loop back into the smoothing state. Alarms in the smoothing state depend on the arrival of other alarms to trigger an exit out of this state. However, the smoothing timer, which is set when the alarm enters the smoothing state, limits the dependence of the alarm on subsequent alarms. Since the smoothing timer is finite, exit from the smoothing state is guaranteed.

- 7) Since the closed state is the sink of the graph, and there is no possible re-entry into the graph, there can be no loop initiated at the closed state. The action triggered at the closed state is completion of the finite state machine, therefore exit from the closed state and from the finite state machine is guaranteed.
- 8) If there were a loop in the finite state machine, there would exist a state in which re-entry of the token from a state further in the progression of the graph is possible. But, no such state exists. Therefore, there is no possible loop in the finite state machine.
- 9) If there was a possible deadlock of communication between finite state machines, there would exist a state in which processing does not guarantee an exit in a finite amount of time. However, all of the actions within every state and all of the triggers that advance the token from one state to the next are guaranteed to occur in a finite time period. Therefore, there is no deadlock between finite state machines.

IV. THE RULE-BASED REASONING ENGINE

The correlation state of the finite state machine incorporates a set of rules based on the finite possible combination of alarms that can occur. The Correlation Module sits at the ALMoM and receives three types of alarms: Provider Network Alarms, User Network Alarms, and Predictive Alarms. The alarm is categorized in the ground state, before arriving at the correlation state. Next, the correlation state checks the link IDs to determine whether there are any related alarms, either from same source or another source (another network layer, or another type of Agent). The Rule-Based Reasoning Engine then has all of the necessary information to assign the next state to the alarm.

Rule-Based Reasoning was chosen for the alarm correlation because of its strengths and their applicability to the Alarm Manager framework. The strengths of Rule-Based Reasoning include simplicity of representation and the explicitness of the rules. Rule-Based Reasoning is best suited for small, well-defined domains. Since the alarms handled by the Alarm Manager are specifically categorized, and there is a finite combination of these alarms, the domain of interest is small and well-defined. Additionally, the possible actions taken based on the combination of alarms are specific and previously determined. The output of the correlation engine is to simply determine the next state of the alarm.

The disadvantage of the Rule-Based Reasoning method is that it does not incorporate a learning mechanism to determine the next state based on a previously addressed set of circumstances, as with the Case-Based Reasoning method. However, since the domain of interest is small and the output of the correlation engine is predetermined, the simplicity of the Rule-Based Reasoning method provides an advantage that is appropriate for the correlation state.

TABLE I
THE RULE BASE FOR THE RULE-BASED REASONING ENGINE.

Rule 1.	Predictive	To Smoothing State
Rule 2.	Predictive-Predictive	To Smoothing State
Rule 3.	Predictive-Predictive-Predictive	To Reconfiguration State; Other Predictive Alarms to Closed State
Rule 4.	Predictive-Provider	To Closed State
Rule 5.	Predictive-User	To Closed State
Rule 6.	Provider	To Recovery State
Rule 7.	Provider-Provider	To Hold State
Rule 8.	Provider-Predictive	To Recovery State; Predictive Alarm to Closed State
Rule 9.	Provider-User	To Hold State
Rule 10.	User	To Recovery State
Rule 11.	User-User	To Hold State
Rule 12.	User-Predictive	To Recovery State; Predictive Alarm to Closed State
Rule 13.	User-Provider	To Hold State

A. Components of the Rule-Based Reasoning Engine

A Rule-Based Reasoning Engine contains a working memory, a rule base, and a reasoning algorithm. The working memory contains the topological and state information of the system, and recognizes network alarms. For the Alarm Manager system, the working memory needs to know the progression of an alarm in the finite state machine. Alarms are given directly to the working memory upon invocation of the correlation state. The rule base contains rules in the form of *condition-action*. The reasoning algorithm compares the current state of the system with the rules in the rule base, and tries to find the closest match in order to produce output.

B. The Correlation Rules

It was determined that, after the alarms are categorized, 13 possible combinations of alarms must be handled by the correlation state. These 13 cases each have an outcome that is individually defined in the rule base.

Table I gives the rules in the rule base, defined as a combination of categorized alarms and the actions resulting from each combination. Each combination of alarms is connected to the same physical link, as determined by a comparison process in the correlation state. The first alarm in each rule is the most recently received, and the action in the next column is related to this alarm unless otherwise stated.

There are other possible combinations of alarms not explicitly stated in the rule base. For example, a User-User-Provider combination is possible, and it will take the same action as a User-Provider combination. In other words, if there is a working Provider Alarm, any related User Alarms that arrive will take the same action: advance to the hold state. A similar explanation applies to any other combinations of alarms not present in the rule base.

As an example, figure 4 shows a time line of alarms received and the resulting actions according to the rules listed in table

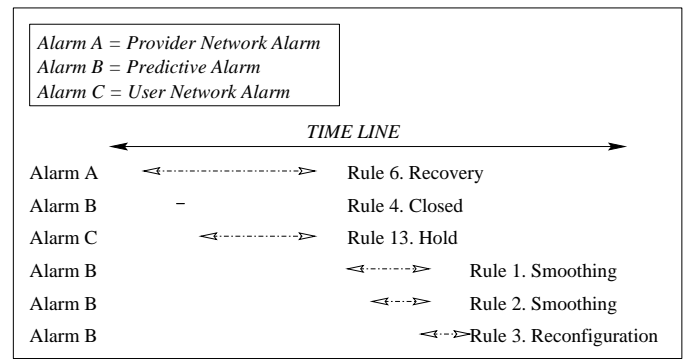


Fig. 4. An Example Time Line of Alarms and Corresponding Rules

I. In this figure, there are six total alarms received in the specified time period. Since all of these alarms refer to the same physical link, the correlation state determines that they are related. When the first alarm, a Provider Network Alarm, is received, there are no active related alarms in the system. Therefore, Rule 6 states that this alarm should go to the recovery state. Since the physical link has triggered an alarm, when the next alarm, a Predictive Alarm, is received, it is dismissed according to Rule 4. It is logical that bandwidth should not be adjusted during a time period when traffic on a link is being re-routed. The next alarm, a User Network Alarm, advances to the hold state according to Rule 13. This alarm remains in hold state until recovery of the Provider Network Alarm is complete, at which time both Alarm A and Alarm C advance to Closed State. Subsequently, a Predictive Alarm is received. Figure 4 shows that when three Predictive Alarms are received, the third alarm goes to reconfiguration state. When the third Predictive Alarm arrives into the system, the first two Predictive Alarms advance from smoothing state to closed state.

V. CONCLUSION

In this work, we have presented a multiple-layer alarm management approach that can help improve the efficiency of a layered network's performance in terms of fault management and performance management.

We have developed a framework for a multi-layer Alarm Manager that incorporates both the temporal and spatial aspects of alarm management. Two mechanisms have been defined to represent the processing of the Alarm Manager: a finite state machine, and a set of rules for a rule-based reasoning engine.

For further study, it would be interesting to develop a more extensive correlation module that could handle a broader range of alarms. That is, the issue of alarm categorization at the Network Manager level in this system could be examined. Another possible extension of this study would be to replace or enhance the rule-based reasoning engine so that it incorporates learning into the system. The simple smoothing state could also be extended to incorporate a more complex smoothing logic. The issue of scalability of the alarm manager in the

event of an alarm storm could be further examined. In addition, more simulations are needed, including simulations on large-scale multi-domain networks, for a further level of validation of the alarm management framework.

REFERENCES

- [1] A. T. Bouloutas, S. Calo, and A. Finkel, "Alarm Correlation and Fault Identification in Communication Networks," *IEEE Transactions on Communications*, Vol. 42, pp. 523-533, 1994.
- [2] G. Jakobson, G., and M. D. Weissman, "Alarm Correlation," *IEEE Network*, Vol. 7, pp. 52-59, 1993.
- [3] G. Jakobson, M. Weissman, L. Brenner, C. Lafond, and C. Matheus, "GRACE: building next generation event correlation services," *Network Operations and Management Symposium, 2000*, pp. 701 -714, 2000.
- [4] L. Lewis, "Event Correlation in SPECTRUM and Other Commercial Products." July 1999, available from <http://63.127.194.2/literature/white-papers/wp0551.pdf>.
- [5] D. Medhi, S. Jain, D. Shenoy Ramam, S. Thirumalasetty, M. Saddi, and F. Summa, "A Network Management Framework for Multi-Layered Network Survivability: An Overview," *Proceedings of IFIP/IEEE Conference on Integrated Network Management (IM'2001)*, Seattle, WA, pp. 293-296, May 2001.
- [6] M. Subramanian, *Network Management Principles and Practice*. Addison-Wesley, Reading, Massachusetts, 2000.
- [7] M. Thottan, and C. Ji, "Proactive Anomaly Detection Using Distributed Intelligent Agents," *IEEE Network*, Vol. 12, pp. 21-27, 1998.