RISK AND RELIABILITY ASSESSMENT FOR TELECOMMUNICATIONS NETWORKS*

Gregory D. Wyss
Risk Assessment & Systems Modeling
Sandia National Laboratories
Albuquerque, NM 87185-0747
(505) 844-5893

Heather K. Schriner
Risk Assessment & Systems Modeling
Sandia National Laboratories
Albuquerque, NM 87185-0747
(505) 844-2719

Timothy R. Gaylor
Data Transport & Network Design
Sandia National Laboratories
Albuquerque, NM 87185-0451
(505) 844-8228

ABSTRACT

Sandia National Laboratories has assembled an interdisciplinary team to explore the applicability of probabilistic logic modeling (PLM) techniques to model network reliability for a wide variety of communications network architectures. We have found that the reliability and failure modes of current generation network technologies can be effectively modeled using fault tree PLM techniques. We have developed a "plug-and-play" fault tree analysis methodology that can be used to model connectivity and the provision of network services in a wide variety of current generation network architectures. We have also developed an efficient search algorithm that can be used to determine the minimal cut sets of an arbitrarily-interconnected (non-hierarchical) network without the construction of a fault tree model. This paper provides an overview of these modeling techniques and describes how they are applied to networks that exhibit hybrid network structures (*i.e.*, a network in which some areas are hierarchical and some areas are not hierarchical).

I. BACKGROUND

For many years, probabilistic logic modeling (PLM) techniques have been used to help assess the reliability of complex electro-mechanical systems ranging from individual components within automobiles to large precision machine tools and even complex semiconductor fabrication facilities. Related techniques have been used to assess the risks associated with potentially high-consequence facilities such as chemical processing plants and nuclear power reactors. These techniques provide designers and analysts with key insights that can be used to predict the most important failure modes and vulnerabilities in the system. They can also provide quantitative guidance as to the most cost-effective ways to improve overall reliability.

While PLMs have been commonly used in many industries, their use in the telecommunications industry has been fairly limited. The complex topologies of communications networks and the time-dependent interactions between network elements[1] have been difficult to model with the fault tree, event tree, and reliability block diagram models that have proven so successful in other industries. However, network designers and analysts could benefit greatly from the quantitative risk and reliability guidance that these types of models can yield.

An interdisciplinary team has been assembled at Sandia National Laboratories to examine how PLM techniques might be applied to model network reliability for typical current and future generation communications network architectures. This team has found that many current generation network technologies are generally deployed in hierarchical architectures. These hierarchical architectures can be readily modeled using PLM techniques. However, current generation telephone networks as well as Asynchronous Transfer Mode (ATM) data networks are often deployed in non-hierarchical ("flat") topologies, and non-hierarchical networks are very difficult to model using fault tree methods. Because of these observations, we have developed two separate algorithms for finding risk and reliability information (cut sets): a "plug-and-play" fault tree analysis methodology for modeling hierarchical networks, and an efficient search algorithm for modeling non-hierarchical networks. We have also shown that these algorithms can be "married" by cut set substitution in order to provide insights for hybrid hierarchical/non-hierarchical networks.

* This work was performed under the Laboratory-Directed Research and Development Program at Sandia National Laboratories. Sandia National Laboratories is operated by Sandia Corporation, a Lockheed Martin Company, for the U.S. Department of Energy under contract DE-AC04-94AL85000.

II. HIERARCHICAL NETWORK ANALYSIS

A network can be said to be hierarchical if either the network address space or the physical structure of the network architecture enforce a hierarchy. In such a network there are usually only a few paths from one node to another. Because such networks contain few redundancies, they are less expensive to construct and easier to manage than their non-hierarchical cousins. Many existing data networks are hierarchical. Examples include most corporate data networks, much of the "Internet", and most cable television distribution and data networks. In addition, some non-hierarchical networks exhibit characteristics that make them *behave* almost hierarchically. For example, although "911" emergency services are provided on the non-hierarchical public telephone network, these services often behave hierarchically, with the top of the hierarchy being the public service answering point. While the following discussion assumes a rigid hierarchy within the network, the method can accommodate a limited number of "cross-cuts" through the hierarchy without becoming overly burdensome.

The PLM method chosen for analyzing communications networks is fault tree analysis (FTA).[2] Fault trees were developed and solved in a modular fashion for both network devices and network architectures. This enabled us to develop a "plug-and-play" fault tree analysis methodology that allows a person with network experience but little FTA experience to successfully model most hierarchical network architectures.

A. Models for Devices

Sandia's research started by building fault trees to assess the modes by which individual network devices can either fail to communicate with the network or disrupt traffic on the network for other users. One of the primary prerequisites to building these fault trees was deciding what types of failure modes[3] to include in the models. Some types of failures are obvious, such as a cable break or someone unplugging the power cord. Other failure modes, such as network interface beaconing, are less obvious. After examining many candidate failure modes, we decided on a representative set of failures to be incorporated into our model of legacy network devices. These failures were then used to build fault trees to represent a network constructed with these devices and, ultimately, to construct the fault tree modules that represent these devices in the "plug-and-play" methodology.

B. Models for Network Architectures

Once the failure modes were developed for the individual network devices, the next step was to define what constitutes success and/or failure for the network as a whole. A few possible metrics for measuring network success are:[4] (1) Device A can communicate with Device B; (2) minimum bandwidth requirements for all users are met; (3) isochronous (video, voice) data arrives at the destination in time to be useful; (4) switches, routers, etc., are not saturated and do not have to discard data; (5) errors are at a minimum and do not significantly affect user performance; (6) all users that need a particular service can gain access to it within a reasonable amount of time; and (7) network security is maintained at all times.

One immediately notices that some of these metrics contain fuzzy terms such as "reasonable" and "in time" and "useful." How, for example, does one define "a reasonable amount of time" for access to a particular network service? A "reasonable" time period for a Cray supercomputer is likely to be very different from that for an 80386-based PC. Similarly, the definition of "in time" for data that controls a weapons system on a battleship that is under attack may be quite different from that for an engineer remotely accessing a CAD package over the network. So what are the criteria for measuring a failure quantitatively? When does poor performance constitute a failure? The answers to these questions vary for each network depending on its design purpose. Human health and safety, business productivity, and corporate profits must all be considered when defining success for a particular network. Therefore, the success metrics that are chosen for each network must be appropriate for the applications that the network is intended to implement. Furthermore, the success metrics must be reevaluated over time as the mission of the network evolves.

While the above criteria are very flexible, they still do not present an adequate definition for network failure. Qualitatively, the user of device A will perceive that the network has failed whenever they cannot communicate with any needed device B. The user will also perceive that the network has failed if a needed network service is unavailable for a measurable amount of time. These qualitative observations by users are valid and important even though they are unlikely to represent the best *quantitative* measures of network success as described in the previous paragraph. Therefore, since they are generic to all networks and germane to the users' perception of network failure, the following metrics of network success were chosen as a starting point for our fault tree analysis of local area networks:

1. Within each workgroup (or sub-network), all devices (routers, workstations, etc.) attached to the network can intercommunicate with one another (the "local connectivity" condition).

2. Bridges and routers that interconnect workgroups on different LAN segments need to be operational for normal day to day operation (the "global connectivity" condition).

3. Network services must be functional for the network users that depend upon them (file servers, Novell servers, mail servers, etc.).

We began to model network architectures by constructing fault tree models for the local connectivity condition. We developed fault tree models for most current types of local networks, including individual device-to-device links, as well as ethernet, token ring, and FDDI architectures (the arbitrary interconnectivity of ATM networks was handled differently, and is discussed later). We demonstrated that it is a straightforward exercise to construct fault tree connectivity models for each of these classes of networks, and that the resulting cut sets do not contain any features that would make them incompatible with the traditional cut set importance measures described previously.

After achieving success modeling local connectivity, we sought to model the global connectivity condition. There are potentially an infinite number of ways that local networks can be combined to form larger corporate and global networks. However, if a physical and/or logical hierarchy is strictly maintained, or if there are not many "crosscuts" through the hierarchy, then one can develop a global connectivity fault tree by starting at the highest point in the hierarchy and working towards the bottom using the same basic techniques that were developed for the local connectivity condition. In this way we developed fault tree connectivity models for a variety of realistic hierarchical network architectures that incorporated many different combinations of networking technology. The fault trees were developed, solved, and analyzed for component importance using existing Sandia risk analysis software.[5,6]

C. Models for Network Services

We have also successfully incorporated the availability of network services into our fault tree connectivity models. In a typical network, these services are provided to network users by one or more server computers. A user is able to use a particular service if all of the following are true: (1) the user's computer can communicate with the network, (2) the server machines that provide the particular service are available and can communicate with the network, and (3) the network is able to carry traffic between all of these machines.

Condition 1 simply requires that the user's machine be in working order, and, since this can be assessed separately from network connectivity and service conditions, individual user machines are generally not incorporated into the network fault tree model. For a simple single-server network service, condition 2 requires only that the server machine be working and communicating with the network. Finally, for a single user of network services, condition 3 is simply a subset of our global connectivity condition. If we look at the availability of network services to *all* users, condition 3 becomes a larger subset of this same global connectivity condition because, while every user must be able to communicate with the server machine, every user need not be able to communicate with every other machine in order to obtain network services. However, the user does not view the network as successful if global connectivity is violated. Thus, in the spirit of modeling network success through the eyes of the user, condition 3 can be replaced by the global connectivity condition without loss of applicability. Therefore, a fault tree that models both network connectivity and network services can be constructed based upon the following success criteria: the network-based information system is successful only if global connectivity is maintained *and* servers are available to provide all necessary network services (thus, from the user's perspective, the network-based information system fails if either the server *or* network connectivity fails). In a network where a single server provides all network services, the applicable fault tree model simply consists of a logical OR condition of the availability of the server machine with the network connectivity model we developed in previous sections. For more advanced networks with multiple and possibly redundant servers, the single server in the OR condition would be replaced by a logical model (likely a small fault tree) that examines the combinations of server machines that must be functional in order for all network services to be available. This fault tree is usually easy to construct given the network specifications.

D. The "Plug and Play" Methodology

As we developed more and more fault tree global connectivity models, we became aware that a user of our methods would be required to have significant expertise in FTA methods in order to assure that their models were in fact implemented correctly. However, in order for FTA methods to be applied widely in the networking community, they must be made accessible to network analysts who are at most casual fault tree analysts. Our objective was to develop a methodology that would allow network designers to construct a fault tree model in much the same way one might think about assembling a network architecture diagram: by simply "plugging together" model elements that represent easily identified network components to, in essence, automatically build the fault tree model. Under a "plug-and-play" modeling technique, an expert constructs generic fault tree "modules" to represent the failure modes of typical network components and subnetwork architectures.[7,8] A casual analyst can then "plug" these modules together to quickly form a complete fault tree global connectivity model for a complex hierarchical network. There are a number of advantages to this approach. By creating individual fault tree modules for each network component which can be combined together to model an overall network, an initial fault tree model of a complex network can be constructed quickly and efficiently. Furthermore, changing network configurations can also be easily incorporated into the model. Finally, the method for building such fault trees will be familiar to many network designers and analysts. This section describes the construction of generic fault tree modules and the method by which they are combined to form a fault tree global connectivity model.

1. Generic Fault Tree Module Development. The universe of available network elements is large and ever-growing. Since our research project is relatively small, and its objective is methodology development (as opposed to setting up a production analysis environment), we chose to model only a representative subset of typical network elements (*e.g.*, traffic routing and switching hardware, and a generic class of end user devices) and architectures (*e.g.*, FDDI rings, token rings, and a variety of ethernet architectures) in generic fault tree modules. Each generic fault tree module must contain all of the important failure modes that its network element may exhibit in any situation. Since not all failure modes may apply to *every* situation, the network analyst will "trim" from the final fault tree model those failure modes that do not apply to the particular situation at hand. Each fault tree module consists of two interconnected parts: (1) a section for the failure modes for the network component itself, and (2) a second section to include failure modes for any attached components. The generic fault tree modules for other attached components are "plugged into" the second section of this network element's generic module under the "plug-and-play" methodology. The generic fault trees were specifically structured so that it would be easy to combine them in arbitrary ways and model any network configuration with ease.

2. Combining the Generic Modules. The first step in applying the generic fault tree modules to model a network is to determine the network element that sits at the top of the network hierarchy. In most cases, the top network element is one whose failure would cause the greatest loss in communication abilities. This network element often also resides at the top of a logical address hierarchy within the network. We select the generic fault tree module for this top network element to be the basis for the overall fault tree for the network (it forms the top of our fault tree model).

The next step in this method is to "reach out" from the top network element toward the bottom of the network hierarchy by attaching the generic fault tree modules for any components that are found along the way. Thus, any components or sub-network architectures that are directly connected to the top element of the network hierarchy are modeled by substituting or "plugging in" the connected component's generic fault tree module into the attachment branch of the top element's fault tree. These newly modeled network elements are then examined to determine the components that are attached to them. As each new network element is identified, its generic fault tree module is "plugged into" the appropriate attachment branches of the emerging fault tree "stem". This process continues until the entire network has been modeled. Once all network elements are included in the fault tree model, any remaining unused attachment branches are simply trimmed off because they represent network attachment options that were not exercised in the current network architecture. At this point the fault tree model is complete and ready to be solved.

Note that the fault tree development process can be broken off before *all* elements are incorporated in the model if the analyst is interested in modeling the characteristics of only a specific portion of the network (say, the network backbone). The analyst can then extend this fault tree model to successively lower levels in the hierarchy without any loss of information by simply reviving the appropriate attachment branches and continuing to apply the "plug-and-play" methodology as described previously. The fault tree paradigm naturally supports this concept of a high-level "quick look" followed by iterative model refinement. Since the model can be evaluated at any level of detail, it can provide a relatively inexpensive method for investigating high-level questions about the network. It also provides a cost effective way to play "What if?" games on early network designs as the network designer experiments with different ways to provide maximum reliability for the user community.

III. NON-HIERARCHICAL NETWORK ANALYSIS

Non-hierarchical networks have no enforced physical or logical hierarchy. They are often designed with a high degree of redundancy, so there may be many paths from one node to another. This makes these networks well-suited for use in areas where a high degree of reliability is important. This redundancy is, however, expensive to install and can be difficult to manage. Examples of non-hierarchical networks include the public telephone voice/data network and many ATM data networks.

Fault tree models become extremely difficult to construct for non-hierarchical networks for two reasons. First, there is no hierarchy, and there can be data links between arbitrary combinations of nodes. These links act like "cross-cuts" through a hierarchical network, and even a small number of these "cross-cuts" can greatly increase the difficulty of construction for individual fault trees. Second, to properly model global connectivity ("everyone can talk to everyone") can require the construction of many fault trees in order to appropriately capture all of the possible paths between all combinations of nodes. These factors combine to render FTA cumbersome and ineffective for most realistic non-hierarchical networks.

Previous approaches to modeling the reliability of non-hierarchical networks have focused on path set theory.[9,10] The objective of a path set analysis is to determine all of the possible paths over which data can flow as it travels from one endpoint to another. Path set analysis is an efficient reliability analysis when the analyst is principally concerned with traffic between two or at most a few points in a network. However, to truly model global connectivity (often called the k-terminal network reliability problem) using path sets, one would have to examine the path sets connecting every possible pairwise combination of network endpoints. This is very computationally expensive. Also, path sets are not suited to the types of component importance measurement that are naturally derived from cut sets. Some have noted that cut sets can be derived from path sets through the mathematical principal of duality. While this duality-based approach is theoretically very appealing, it should be noted that the actual determination of duality is computationally very challenging given a physically reasonable number of path sets.

Because of the limitations of these commonly used techniques, Sandia National Laboratories has developed an efficient search algorithm to find the global connectivity cut sets for arbitrarily interconnected networks. This method determines cut sets based on the network connectivity diagram, so there is no need to construct and maintain a separate reliability model. The algorithm takes advantage of a number of architectural and mathematical properties to reduce the computational effort required to obtain global connectivity cut sets for these networks. These cut sets can then be mathematically combined into the OR condition described in a previous section to obtain system cut sets that consider network services.

While the high degree of redundancy that can be present in non-hierarchical networks can result in a highly reliable system, it has a major drawback for risk and reliability analysts: the number of cut sets that are required to model full connectivity can be extremely large. The most computationally expensive portion of the solution involves the direct search of the network for cut sets. We can minimize the work required in that portion of the solution by first simplifying the network prior to solution, and second, by extracting as much information as possible from the cut sets generated during the solution process.

A. Simplifying the Network

One obvious way to reduce the amount of computational effort required to search the network for cut sets is to search through a simpler network. The concept of solving a series of simpler problems and then assembling the results at the end is similar to the concept of independent subtrees (ISTs) in FTA. In FTA, an IST is any portion of the fault tree (a subtree) such that, when it is separated from the remainder of the fault tree, every event in the subtree occurs only within that subtree and not within the remainder of the fault tree (the "stem"). Mathematically, these subtrees are independent of the remainder of the fault tree, so they can be removed, replaced by a placeholder primary event on the stem, solved separately from the stem, and reintegrated with the stem solution at the end of the analysis through variable substitution. The computational savings comes about not only because we are solving several simpler problems instead of a single complex problem, but also because the independence of the ISTs means that we need not perform the computationally expensive Boolean reduction step as the IST cut sets are reintegrated with the stem cut sets.

A similar type of simplification can be accomplished for non-hierarchical networks. First, any node or group of nodes that are attached to the remainder of the network by only a single link or node can be thought of as a "tail" and can be removed as an independent network. It can be solved separately from the remainder of the network, and the cut sets for the entire network consist simply of the cut sets from the tail subnetwork "OR"ed with those from the remainder of the network "OR"ed with the failure of the single link or node that connects these partial networks. Since no link or node can occur in both subnetworks, the two solutions are independent of one another and need not be subjected to Boolean reduction.

A second type of simplification can be performed for any node or group of nodes that are attached to the remainder of the network by only two paths (links or nodes). This subnetwork can be thought of as a "loop" and the network can be subdivided at this attachment point. The loop can be removed and replaced by a dummy link that will be used as a surrogate for the removed subnetwork. The loop and the remaining subnetwork are now solved separately. The final network solution is generated by substituting the cut sets found for the loop into any cut set that contains the dummy link as one would backsubstitute the cut sets found for an independent subtree into the cut sets for the fault tree stem. Again, since no link or node can occur in both subnetworks, the two solutions are independent of one another and need not be subjected to Boolean reduction.

In some cases it is possible to identify subdivision points such as tails and loops automatically with a minimal amount of computational effort. This is generally true when the loop or tail consists of only a single string of nodes. When the singlely- or doubley-connected subnetwork is more complex than this, however, it can take as much computational time to identify the subdivision point as it does to solve the complex problem. Since these network architecture features are often easy for the analyst to identify visually, it is possible to obtain significant computational savings even when automated network subdivision methods fail.

B. Inference of Cut Sets

The most computationally expensive portion of the solution is directly searching the network for cut sets. Therefore, we would like to spend as little time as possible in this portion of the analysis and infer as much information as possible from its results. If possible, we would like to search out only some fraction of the actual cut sets and infer the existence of the remaining cut sets from the search results.

If we pause for a moment at this point to look at the reason that a network exists, we can derive insight that will help us as we seek to minimize network searching activities. A network exists for the purpose of transmitting data over links from one point to another. Any failure of communication can be made to appear equivalent to the failure of some set of communication links — regardless of whether the failure actually occurs in a link, a node, or some combination thereof. Therefore, we can without loss of generality search the network for cut sets that contain only link failures (*i.e.*, assume for the purposes of the search engine that the nodes are completely reliable), and then infer in a later non-search operation the existence of cut sets that contain combinations of link and node failures. This inference occurs as a mathematical transformation based on a simple physical principle: there can be no data traffic on a particular link (*i.e.*, the link appears to be failed) if *either* the link itself is failed *or* the node at either end of the link has failed. Therefore, we can use the search algorithm to find those cut sets that contain only link failures, and then expand each link failure event to include the potential for one of its endpoint nodes to cause the link functionality to fail. In this way, a single cut set that contains n links becomes a Boolean expression that, when expanded, can become up to 3^n system cut sets that contain both links and nodes. Unfortunately, since a single node may terminate many links within the same link-based cut set, the resulting 3^n system cut sets may not be minimal. Furthermore, since the failure of a single node can cause the functionality of many links to fail, the fact that we start out with minimal link-based cut sets from the search algorithm does not imply that all of the expanded system cut sets will be minimal. Therefore, the expanded system cut sets must be subjected to Boolean reduction. This is most efficiently accomplished as a three step process:

1. Expand each link-based cut set into its 3^n system cut sets (based on both links and nodes), and perform Boolean reduction on the resulting equation,

2. Assemble all of the remaining reduced system cut sets into a single cut set expression, and

3. Perform a Boolean reduction across all system cut sets to obtain the minimal system cut sets for the overall network.

It should be noted at this point that the resulting minimal system cut set expression is likely to be very large (many thousands of cut sets) for any realistic non-hierarchical network. Thus, it is very important that network simplification be performed before either the network search or cut set inference steps are performed. The independence of subnetworks described in the previous section means that the recombination of subnetworks can occur after the cut set inference step, thus further reducing the computational effort associated with the Boolean reduction operations.

Ongoing research indicates that a further dramatic reduction in computational effort may be possible for these classes of problems. It may be possible to use the link-based cut sets to directly construct system cut sets that are *minimal* both within a single link-based cut set *and* over the entire group of link-based cut sets. This would mean that all of the previously mentioned Boolean reduction steps could be avoided — a major improvement in computational efficiency. The discussion of that technique, should it prove viable, will be the subject of another paper.

C. Search Algorithm

Given the constraints described in the preceding sections, we are now ready to discuss the search algorithm that is used to obtain link-based cut sets for each of the identified subnetworks. The algorithm begins with the arbitrary selection of a starting node within the network. We will then "reach out" from that node incrementally to find all possible ways to subdivide the network, and the link-based cut set associated with each subdivision.

The first step in the search algorithm is to arbitrarily select a starting node within the network. We then construct the link-based cut set that subdivides the network into one subnetwork that contains only the single starting node, and one subnetwork that represents the remainder of the network. This cut set is trivial to construct — it simply consists of the list of all links that are attached to the single node.

The next step in the network is to reach out from our starting node to make a subnetwork that contains two nodes: our starting node plus one other node to which our starting node is attached (via a link). We again construct the link-based cut set that subdivides the network into one subnetwork that contains only our two selected nodes, and one subnetwork that represents the remainder of the network. We repeat this process until we have link-based cut sets for all possible two-node subnetworks that contain our starting node. After completing the cut sets for two-node subnetworks, we perform the same process for three-node subnetworks, then for four-node subnetworks, and so on until, for an *n*-node network, we have completed this process for all possible *(n-1)*-node subnetworks. At that point we will have generated the link-based cut set for every possible subdivision of the network, and the algorithm terminates.

IV. APPLICATIONS

The techniques described in this paper can be used both to obtain network connectivity cut sets and to assess the potential for particular types of equipment failure to affect overall network performance. We have successfully used these techniques to assess the reliability of and/or the risks associated with a variety of types of hierarchical and non-hierarchical network-based information systems, including enhanced 911 emergency services architectures, public telephone common channel signaling networks, non-hierarchical data networks (LAN/MAN/WAN environments), and high-speed ATM data networks. The importance measures that were derived for these cut sets have been shown to be helpful to network designers in developing more robust network architecture implementations. They have also proven useful to network operations personnel as they seek to prioritize both network upgrade and network maintenance activities. Cut set results can also be used with discrete optimization techniques such as genetic algorithms to help an analyst select the most effective system improvement approach from a possibly large number of candidate strategies.[11,12]

While the techniques that have been used to generate system cut sets for this project are somewhat unique, the cut sets are in every way traditional Boolean cut sets. Thus, they can also be subjected to location transformations as would be typical in an "external events" PRA analysis (*e.g.*, examining the risks associated with fire and flood events) or a "vital area" analysis (for evaluation of security practices). In fact, the term "location" can in this instance refer either to a traditional physical location or to a "virtual" location (a location within a computer or network from which an intentional or accidental attack could be launched).[13] It should be noted, however, that these methods do not constitute a "silver bullet" that will solve all network analysis problems. Many of the problems that exist within computers and networks are inherently time-dependent (*e.g.*, they may only impact the system if they occur in a particular order or within a constrained time window). Although traditional fault tree analyses and cut sets do not adequately capture the time-dependent nature of many of these dynamic events,[14,15] they can prove helpful to simulation-based network analysts because they can help point those analysts toward the most risk-significant scenarios. This has been occurring for years in the nuclear power industry as PRA results have helped prioritize analysis and testing programs.

V. SUMMARY

This paper has presented the results from an interdisciplinary team that was formed at Sandia National Laboratories to explore the applicability of PLM techniques to information systems. We have demonstrated that many aspects of information systems can be modeled using a "plug-and-play" fault tree analysis technique as well as a new efficient cut set search technique for use with non-hierarchical network architectures. We have also demonstrated that the types of results that can be obtained from PLMs can be of great practical value to network designers as well as operations personnel. These PLM techniques are intended not to replace current network analysis methods, but to supplement them. They provide additional tools for the network designers' workbench to enhance their depth of understanding so that they can design more functional and reliable network systems.

REFERENCES

1. M.O. Ball, "Computational Complexity of Network Reliability Analysis: An Overview," *IEEE Trans. Reliability*, Vol. R-35, No. 3, pp. 230-239, August 1986.

2. N.H. Roberts, W.E. Vesely, D.F. Haasl, and F.F. Goldberg, "Fault Tree Handbook," NUREG-0492, U.S. Nuclear Regulatory Commission, Washington, D.C., January 1981.

3. "Novell's Guide to Netware LAN Analysis," 2nd Edition, Novell Press, 1993.

4. A.S. Tanenbaum, "Computer Networks," 2nd Edition, Prentice Hall, New York, 1991.

5. B. Bingham, J. Hutchison, and B. Lopez, "SEATREE Version 2.62 User Manual," prepared for Sandia National Laboratories by Science and Engineering Associates, Albuquerque, New Mexico, 1994.

6. K.M. Hays, G.D. Wyss, and S.L. Daniel, "A User's Guide to SABLE," Sandia National Laboratories, Albuquerque, New Mexico, 1996.

7. G.B. Varnado, W.H. Horton, and P.R. Lobner, "Modular Fault Tree Analysis Procedures Guide," SAND83-0963, NUREG/CR-3268, Prepared by Sandia National Laboratories for the U.S. Nuclear Regulatory Commission, Washington, D.C., August 1983.

8. T.L. Zimmerman, N.L. Graves, A.C. Payne, and D.W. Whitehead, "Microcomputer Applications of and Modifications to the Modular Fault Trees," SAND89-1887, NUREG/CR-4838, Prepared by Sandia National Laboratories for the U.S. Nuclear Regulatory Commission, Washington, D.C., June 1990.

9. T. Polif and A. Sathyanarayana, "Efficient Algorithms for Reliability Analysis of Planar Networks — A Survey," *IEEE Trans. Reliability*, Vol. R-35, No. 3, pp. 252-259, August 1986.

10. R.K. Wood, "Factoring Algorithms for Computing *K*-Terminal Network Reliability," *IEEE Trans. Reliability*, Vol. R-35, No. 3, pp. 269-278, August 1986.

11. D.W. Coit & A.E. Smith, "Use of a Genetic Algorithm to Optimize a Combinatorial Reliability Design Problem," Proceedings of the Third IIE Research Conference, 467-472, 1994.

12. L. Painton & J. Campbell, "Genetic Algorithms in the Optimization of System Reliability," IEEE Transactions on Reliability, Special Issue on Design, **44**(2), 172-178, 1995.

13. G.D. Wyss, S.L. Daniel, H.K. Schriner, and T.R. Gaylor, "Information Systems Vulnerability: a Systems Analysis Perspective," Presented at the 12th Annual Security Technology Symposium and Exhibition, American Defense Preparedness Association, Williamsburg, VA, June 17-20, 1996.

14. J.B. Dugan, S.J. Bavuso & M.A. Boyd, "Dynamic Fault Tree Models for Fault Tolerant Computer Systems," IEEE Transactions on Reliability, **41**(3), 363-377, 1992.

15. L.L. Pullum & J.B. Dugan, "Fault Tree Models for the Analysis of Complex Computer Systems," Proceedings of the 1996 Reliability and Maintainability Symposium, 1996.