CONF-9610171--1
SAN096-2048C
SAND--96-2048C

Probabilistic Logic Modeling of Network Reliability for Hybrid Network Architectures

Gregory D. Wyss, Heather K. Schriner and Timothy R. Gaylor
Sandia National Laboratories
P.O. Box 5800, Mail Stop 0747
Albuquerque, New Mexico 87185-0747 USA
gdwyss@sandia.gov

Abstract

Sandia National Laboratories has found that the reliability and failure modes of current-generation network technologies can be effectively modeled using fault tree-based probabilistic logic modeling (PLM) techniques. We have developed fault tree models that include various hierarchical networking technologies and classes of components interconnected in a wide variety of typical and atypical configurations. In this paper we discuss the types of results that can be obtained from PLMs and why these results are of great practical value to network designers and analysts. After providing some mathematical background, we describe the "plug-and-play" fault tree analysis methodology that we have developed for modeling connectivity and the provision of network services in several current-generation network architectures. Finally, we demonstrate the flexibility of the method by modeling the reliability of a hybrid example network that contains several interconnected ethernet, FDDI, and token ring segments.

1. Background

For many years, probabilistic logic modeling (PLM) techniques have been used to help assess the reliability of complex electromechanical 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 components [1] have been difficult to model with the fault tree, event tree, and reliability block diagram models that have been used successfully in other industries. However, network designers and analysts could benefit greatly from the information that PLMs can yield.

An interdisciplinary team at Sandia National Laboratories has found that, in general, current-generation network technologies can be modeled using PLM techniques. We have developed a "plug-and-play" fault tree analysis methodology for modeling connectivity and the provision of network services in a wide variety of current-generation network architectures (*e.g.*, ethernet, token ring, and FDDI) containing various types of components (*e.g.*, routers, concentrators, MAUs, CAUs, servers, and workstations) interconnected in a wide variety of typical and atypical configurations. This paper describes that modeling technique and illustrates why the results that can be obtained from PLMs are of great practical value to network designers and analysts.

2. Benefits of Probabilistic Logic Models

PLMs have been used by a number of different disciplines, including quantitative reliability analysis (QRA), probabilistic risk analysis (PRA), and probabilistic safety analysis (PSA). Regardless of the discipline, the reasons for developing a PLM are the same: to identify an exhaustive list of the modes by which a system can fail, to find an approximate frequency with which we might expect to observe failures, and to determine a rank ordering of the components in the system by their "importance" to the proper function of the system. The "importance" of a component can be defined in a number of ways, but is often thought of as answering one of the following questions:

- How sensitive is the overall system reliability to changes in the reliability of each individual component?
- If the reliability of this component is allowed to decrease (say, by substituting components of lesser quality), how much will this affect overall system reliability?

DISCLAIMER

- If money is invested to increase the reliability of this component, how much will this affect overall system reliability?

Clearly the answers to these questions cut to the heart of how networks are designed and managed. For example, a PLM analysis might show that a particular concentrator does not contribute significantly to the unreliability of the system, but that it would become a significant contributor if its reliability were allowed to deteriorate. The analysis might also show that, while a particular router seems to be a major contributor to system unreliability, the funds required to replace it might be more effectively spent pursuing several less expensive upgrades. It might also show the opposite. PLM results should not be used as the exclusive basis for design and upgrade decisions because such decisions have intangible aspects that must also be considered. However, PLM results do provide *quantitative* answers to network reliability questions, and these quantitative answers can be used as a legitimate benchmark to get past the "gut feeling" that unfortunately forms the basis for many network design and upgrade decisions. It has also been demonstrated that PLM results are well suited for use in discrete optimization algorithms such as genetic optimization.

The wealth of decision support information provided by PLM models sometimes tempts the uninitiated to view PLM as a "silver bullet" that makes traditional forms of network analysis such as dynamic simulation obsolete. This is most certainly *not* the case. PLM and dynamic simulation are complementary tools which, when used together, provide a more complete view of network performance than either can provide by itself. For example, dynamic simulations are often very computationally expensive, so it will not be possible to simulate each network variation that might be of interest. Insights from PLMs can help direct the simulation analyst to the most important variations so that they can get the most valuable information for the computational effort expended. On the other hand, direct simulation analyses will help PLM analysts to be sure that they have properly established important success criteria within their model. PLM provides a global view of the network and quantitatively leads a designer to options for its betterment, while direct simulation provides detailed information about critical situations within a particular network configuration. Clearly both perspectives are necessary for a complete understanding of the network.

At this point someone usually asks, "You speak of quantitative results, but I have no data. Surely the value of your results cannot be any better than the quality of your data, so how can this be of any benefit to me?" That statement is true *if* you are seeking to predict the absolute reliability of the system (*e.g.*, mean time between failures). However, the most useful result from a PLM is often the rank ordering of components by importance. An accurate rank ordering can be achieved even with relatively little measured reliability data. An analyst can often state with relatively high confidence that component A is "somewhat more likely to fail" than component B, or that a router with internal redundancy would be expected to be "much more reliable" than a workstation. The analyst can create groups of components and failure modes such that all elements in the group have similar failure rates, and then rank these groups to obtain a reasonably accurate set of *relative* reliability data. The rank-ordered results from a PLM are accurate even with only relative data. Thus, it is possible to obtain some of the most useful results from a PLM even in the absence of a great deal of measured reliability data.

3. Fault Tree Analysis Methodology

The PLM method chosen for analyzing communications networks is fault tree analysis (FTA).[2] This section provides a brief background on the mathematics of FTA, followed by a description of how fault trees for both network devices and network architectures were developed and solved. This information lays the groundwork for the discussion of the "plug-and-play" fault tree analysis methodology that will be presented in later sections.

3.1 Mathematical Background

FTA is a deductive method that seeks to determine how individual component failures combine to cause the overall failure of a system. An FTA begins with the selection of a "top event" which is simply the definition of the system failure that is being examined. The FTA continues by finding the immediate, necessary and sufficient conditions (events) for the top event to occur. Next, the immediate, necessary and sufficient conditions that lead to each of these new events are found. This process is applied recursively until each event that remains cannot be further broken down. Events that cannot be further broken down are called primary events, and the fault tree development process is complete when there remain no "leaf nodes" of the fault tree that are not primary events.

To illustrate the above process, consider a well that pumps water to a faucet. The top event, which is our definition of system failure, might be "no water comes out of the faucet." The immediate failures that can cause this event are "the faucet is broken" and "no water is available to the faucet." The immediate failures that can cause water to be unavailable at the faucet are "no water from the pump," which can be caused by "broken pump," "no power to the pump," or "no water available from the well." The progression can be continued by investigating why water is not available from the well, or why power is unavailable to the pump, or why the pump or valve is broken. The decision of how far to go before declaring that something is a "primary event" depends upon the objective of the analysis. One can imagine that the fault tree model constructed by a pump manufacturer might look very different from that of a power supply engineer even though they are modeling the same system because their interests lie in different areas of the system. The pump manufacturer might leave the power

supply as a single event while modeling the pump in great detail. The reverse would likely be true for the power supply engineer. The key is to model the system to a level of detail that is appropriate given the purpose of the analysis.

As the immediate, necessary and sufficient causes of each event are found, we must determine how these causes combine to generate the event itself. If all causes must occur simultaneously for the event to occur, then the FTA logic model is built using a logical "AND" condition ("AND gate"). However, if any one of these causes can generate the event, then the FTA logic model is built using a logical "OR" condition ("OR gate"). While the simple AND and OR conditions handle most situations, other logical combinations are also possible, such as "3 of *n* voting" in control logic, and, in communications networks, "2 nonadjacent failures" for modeling connectivity within a self-healing ring-based architecture such as FDDI. This logic forms an integral part of the fault tree model.

The completed fault tree is solved by repeated application of the laws of Boolean algebra. Each gate in the fault tree can be thought of as a small Boolean equation. Thus, the entire fault tree represents a large system of Boolean equations. Mathematical substitutions can reduce this system of equations to a single large Boolean equation that represents all of the ways that primary events can be combined to cause system failure. The laws of Boolean algebra are then applied to remove redundant terms from the equation. After simplification, each combination of events that is sufficient to cause system failure is called a cut set. If each event in the cut set is also *necessary* in order for system failure to be achieved, then the cut set is said to be minimal (its failures are both necessary and sufficient to cause system failure). The complete list of minimal cut sets theoretically represents all of the possible ways that primary events can combine to cause system failure. Practically speaking, there are often far too many minimal cut sets for an analyst to readily examine, so the cut sets are ranked by size and/or probability, and those cut sets with the lowest rank are eliminated.

The complete list of cut sets represents an important result for any PLM. Quantifying this list provides the overall probability of system failure. A ranking of the cut sets by probability shows the most likely failure scenarios for the system. A designer can use this information to design system improvements that remove the most likely failure scenarios. However, there is much more information buried in this list of cut sets. A simple mathematical transformation of the cut sets provides the "importance measures" described previously. The partial derivative of this list with respect to each primary event shows how quickly the reliability of a system will change given variations in reliability of each component. Setting a primary event's failure probability to 1.0 shows how the reliability of the system will be affected if a very low quality component is used for this function. Finally, setting a primary event's failure probability to 0.0 shows the maximum reliability improvement that could be obtained by "fixing" this component. If this value is large, then it may be appropriate to invest money to improve this component. Thus, the list of cut sets is the key that unlocks all of the other valuable information that can be found through FTA.

3.2 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 are not so obvious. After examining many candidate failure modes, we decided that our model of legacy network devices (i.e., non-ATM) should include those failure modes listed in Table 1. These failures were then used to demonstrate how an analyst would build a fault tree to represent a network constructed with these devices.

3.3 Models for Network Architectures

Once the failure modes were developed for the network devices, the next step was to define what constitutes success or failure for the entire network . A few possible metrics for measuring 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 a 486-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.

Device	Single wire Ethernet	Ethernet from a Hub ("star")	Token Ring (with redundant cabling)	FDDI Ring ("self-healing")
Cable	Device(s) attached to cable is off the net	Device(s) attached to cable is off the net	Device(s) attached to cable is off the net and the ring wraps	Device(s) attached to cable is off the net and the ring wraps
Network Interface Card (NIC)	1. Dead (a) 2. Chattering (b)	1. Dead 2. Chattering	1. Swallows token or data. 2. Dead 3. Chattering 4. Beacons	1. Swallows token or data. 2. Dead 3. Chattering
Hub or concentrator	Not Applicable	1. Complete failure (net is down) 2. Partial failure (devices on failed ports are off net) 3. Partial failure with multiple devices attached to each port (network may become segmented)	1. Bypass relay(s) fail in open state (any additional network problem could cause ring to wrap). 2. Bypass relay(s) fail in closed state (device(s) on this segment is off the net)	1. Complete failure (ring wraps and devices on concentrator are off net) 2. Internal failure (ring may not wrap; however, devices on concentrator are off net) 3. Partial failure (devices on failed ports are off net)
Router	1. Port failure (same as NIC failure, also connection to outside world lost) 2. Dead (connection to outside world lost) 3. Misrouted data 4. Dropped packets	1. Port failure (same as NIC failure, also connection to outside world lost) 2. Dead (connection to outside world lost) 3. Misrouted data 4. Dropped packets	1. Port failure (same as NIC failure, also connection to outside world lost) 2. Dead (connection to outside world lost) 3. Misrouted data 4. Dropped packets	1. Port failure (same as NIC failure, also connection to outside world lost) 2. Dead (connection to outside world lost, also the ring wraps) 3. Misrouted data 4. Dropped packets
Bridge	1. Port failure (same as NIC failure, network segments) 2. Dead (network segments)	1. Port failure (same as NIC failure, network segments) 2. Dead (network segments)	1. Port failure (same as NIC failure, network segments) 2. Dead (network segments)	1. Port failure (same as NIC failure, network segments) 2. Dead (network segments)
Ethernet Switch	Same as Router	Same as Router	Not Applicable	Not Applicable

Notes:
(a) Dead - The device is inoperable in such a way that it does not affect other devices on the network.
(b) Chattering - Device constantly sending data when it is not its turn to transmit on the network.

Table 1. Failure Modes for Network Devices.

While the above criteria are very flexible, they still do not present an adequate definition for network failure. Qualitatively, users of device A will perceive that the network has failed whenever they cannot communicate with any needed device B. Users 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) All devices (routers, workstations, etc.) attached to the network can communicate with one another ("local connectivity" condition); (2) bridges and routers that interconnect workgroups on different LAN segments need to be operational for normal day to day operation ("global connectivity" condition); and (3) network services must be functional for the network users that depend upon them (file servers, Novell servers, mail servers, etc.).

Our fault tree modeling of network architectures started by modeling 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 at the conclusion of this paper). 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, the vast majority of these networks are organized as hierarchies — both physically (through the way that subnetworks are interconnected) and logically (through the assignment of network addresses). Most corporate networks and even the Internet are set up in this manner. If a 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 toward 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 included various combinations of networking technology. The fault trees were developed, solved, and analyzed for component importance using existing Sandia risk analysis software.[5],[6]

Recall that a fault tree for a hierarchical network is developed by starting at the top of the hierarchy and working to successively lower levels until all levels and elements in the network are included. Suppose, however, that the fault tree development process was to be stopped at some relatively high level in the hierarchy. The resulting fault tree would be quick to develop, and its solution would examine the reliability of the high-level (often called "backbone") network. The analyst can then extend this fault tree model without any loss of information to successively lower levels until it contains the level of detail required to answer the questions that are important on that particular day. 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 designer experiments with different ways to provide maximum reliability to the user community.

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 PLM methods in order to ensure that the models were in fact implemented correctly. Our goal was to develop a methodology that would allow a network designer with minimal PLM expertise to construct a fault tree model by simply "plugging together" parts of fault trees that were developed by PLM experts to represent easily identified parts of typical networks. A designer could then quickly plug these fault tree "modules" together based on a network architecture diagram to build a complete and high-quality fault tree for any arbitrary hierarchical network. This "plug-and-play" fault tree analysis methodology is the focus of the remainder of this paper.

We have also successfully incorporated the availability of network services into our fault tree connectivity models. However, a detailed description of the method for modeling network services cannot be provided in this paper due to space constraints. That method will be described briefly near the conclusion of this paper.

4. Modeling Methodology

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 fault trees for each network component that can be combined to model an overall network, initial fault tree models can be constructed quickly and efficiently. Furthermore, changes in network configurations can also be easily modeled. 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.

4.1 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 (FDDI rings, token rings, routers, concentrators, MAUs, CAUs, ethernet hubs, and a generic class of end-user devices) 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 apply to *every* situation, the network analyst will "trim" from the final fault tree model

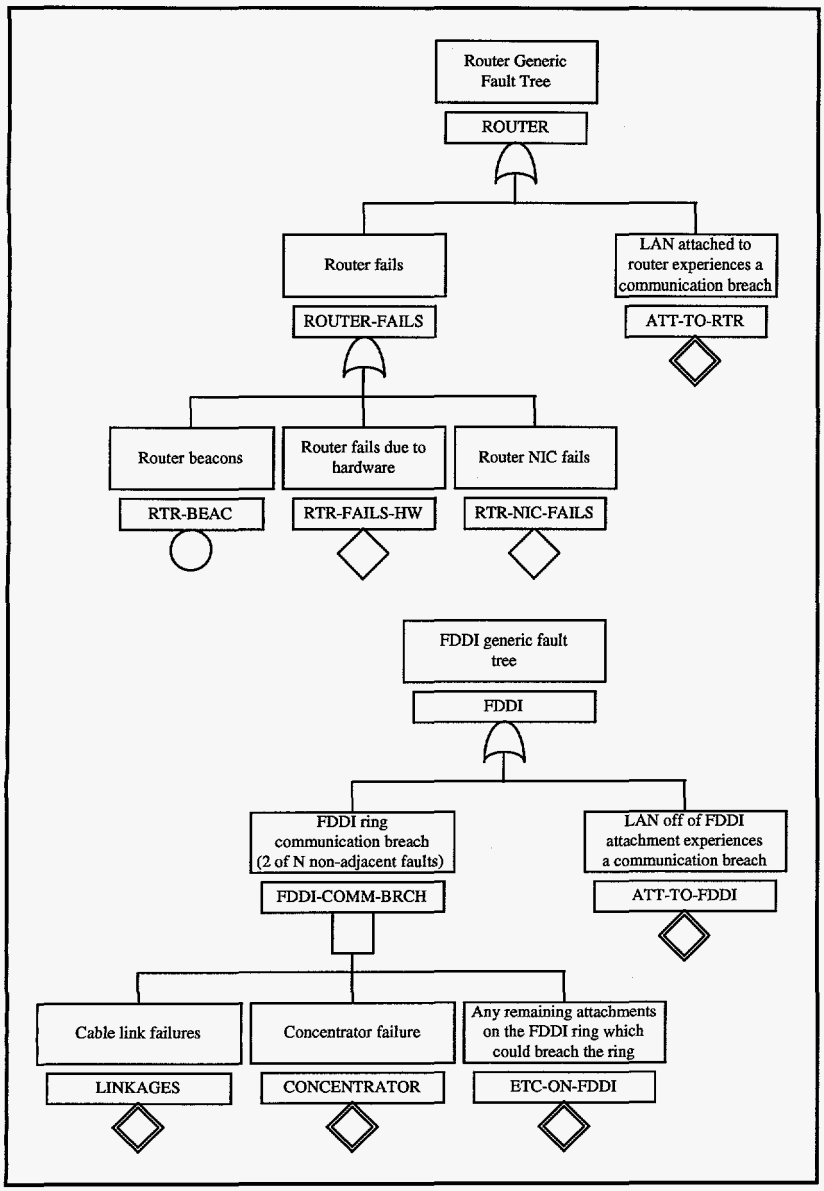

Figure 1. Example Generic Fault Tree Modules.

those failure modes that do not apply to the situation at hand. Each fault tree model consists of two 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. 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 structured to make it easy to combine them in arbitrary ways and model any network configuration with ease.

Simple generic fault tree modules for a router and an FDDI ring are shown in Figure 1. The symbology for a fault tree is as follows: a circle represents a "basic event" (a fundamental component failure), while a single diamond represents an "undeveloped event" (a type of failure that *could* be further modeled by extending fault tree, but is beyond the interest of the current analysis). The double diamond represents a "developed event" and is a point at which other generic fault tree modules would be attached if the network architecture shows other elements attached to this component. The symbol with two upward rounded strokes represents a logical OR combination of events (an "OR gate"), while the square represents some special logical combination other than a simple AND or OR (a "special gate" — in this case, a Boolean equation to represent two nonadjacent failures in an FDDI ring architecture). These sample modules clearly show the two sections (component failures and attached components) described earlier as being typical of these generic fault tree modules.

4.2 "Plug-and-Play" Methodology

The first step in applying the generic fault tree modules to model a network is to determine which network element 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 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 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 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.

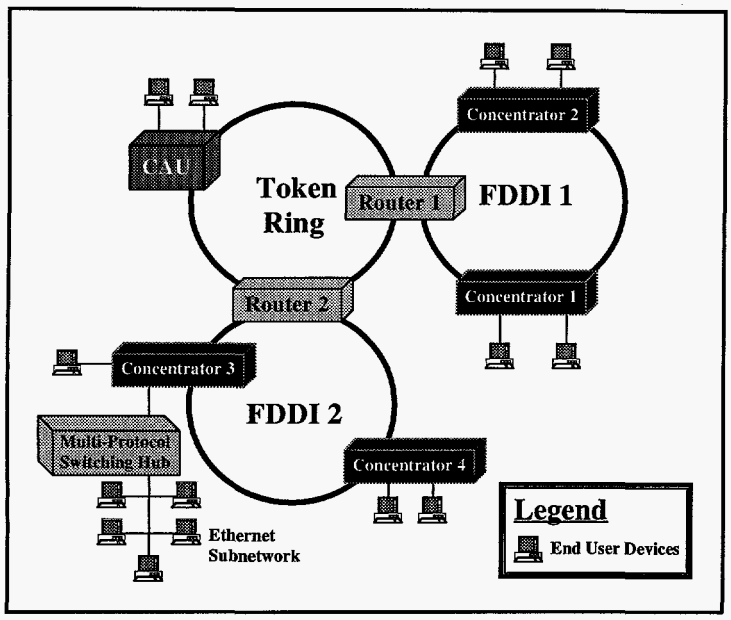

Figure 2. Sample Network Configuration.

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. This provides the basis for iterative model refinement, since the model can be evaluated at any appropriate level of hierarchical detail.

5. Example Problem

5.1 Fault Tree Development

To illustrate the plug-and-play fault tree development process, consider the simple example network shown in Figure 2. The first step in the model construction process is to locate the top point of the network hierarchy. In the example network in Figure 2, Router 1 is chosen to be the top of the hierarchy. The generic fault tree module for a router is selected to be the top of the network fault tree global connectivity model. This can be seen by the top block in Figure 3.

The next step is to determine which network elements are attached to Router 1, and plug generic fault tree modules for them into the attachment branch of the Router 1 fault tree. Based on the network architecture in Figure 2, we attach fault tree modules for an FDDI ring (for FDDI 1) and a token ring to the router fault tree.

The method continues by examining the FDDI 1 and token ring subnetworks. The concentrators on FDDI ring 1 are only attached to end-user devices and not to other LAN segments. Thus, concentrator failures cannot affect any other subnetwork. They can only cause network failure by failing the FDDI ring (or by isolating end user devices, which are not modeled in this example). Therefore, to complete the first FDDI fault tree, the attachment branch is removed ("trimmed") from the generic fault tree modules for Concentrators 1 and 2, and concentrator failure modes (the remaining portions of the concentrator fault tree modules) are plugged into the FDDI ring 1 "concentrator failures" branch. As there are no further attachments to FDDI ring 1, the attachment branch of its fault tree module is also removed. This can be seen on the FDDI ring 1 branch of Figure 3.

Next let us consider the token ring portion of the network. The CAU on the token ring is only attached to end-user devices and is not attached to other LANs. Thus, CAU failures only affect the network by failing the token ring. Therefore, the CAU fault tree attachment branch is removed from the CAU fault tree, and the CAU failure modes are plugged into the token ring "CAU failures" branch in a manner similar to that used in the FDDI tree above. However, since Router 2 is attached to both the token ring and to another LAN, a generic fault tree module for Router 2 is plugged into the attachment branch of the token ring fault tree module.

At this point only one attachment branch remains open: that of the Router 2 generic fault tree module. Since FDDI ring 2 is connected to Router 2, the fault tree module for FDDI ring 2 is plugged into this attachment branch. One concentrator on FDDI ring 2 is attached only to end-user devices and not to other LANs. This concentrator is modeled in the same manner as the concentrators on FDDI ring 1, and its attachment branch is removed. Since a multiprotocol switching hub forms a connection between Concentrator 3 and an ethernet LAN segment, the generic fault tree module for this switching hub is plugged into the attachment branch of the Concentrator 3 fault tree, and this combined fault tree for the switching hub and Concentrator 3 is then plugged into the attachment branch on the FDDI ring 2 fault tree.

This leaves the switching hub attachment branch as the only open attachment branch in the network. The ethernet subnetwork is attached to the switching hub, but this subnetwork is attached only to end-user devices and not to other LAN segments. Thus, failures within this subnetwork cannot affect any other subnetwork. They can only cause network failure by isolating ethernet end user devices from the remainder of the network. Therefore, ethernet failure modes are plugged into the switching hub "ethernet failures" branch in a manner similar to that described previously for other network elements. As there are no other subnetworks to incorporate into the fault tree model, the lone remaining attachment branch is removed from the switching hub fault

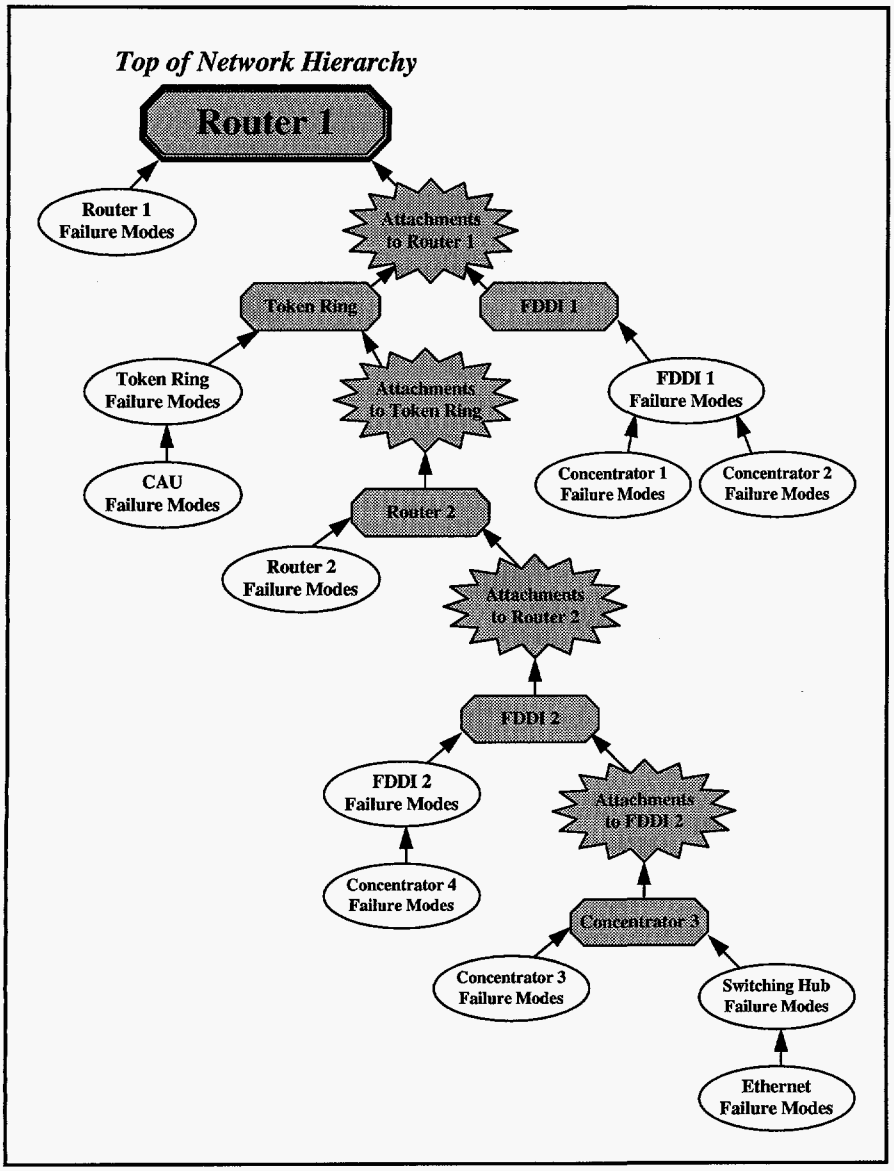

Figure 3. Representation of the "Plug-and-Play" Methodology for the Sample Network.

tree module, and the fault tree is complete. The complete graphical representation of the example is shown in Figure 3. The fault tree itself is not included in this paper due to space constraints.

This example network is simple in order to make it easy to describe the plug-and-play method. It should be obvious, however, that the method can be applied to networks of arbitrary size and complexity by repeating these steps. The only added methodological subtlety arises when a network element such as a router or a switching hub is attached to several (*i.e.*, more than two) subnetworks. If this occurs, one simply replicates the "attachment branches" on the generic fault tree module for that network element so that every attached subnetwork can be represented in the fault tree. Thus, if a router sits on an FDDI ring and attaches three ethernet hubs and two token rings to that FDDI backbone, the attachment branches in the router's generic fault tree module will be replicated to contain five attachment branches (three ethernet hubs and two token rings). This allows all hierarchical networks to be modeled using this plug-and-play methodology.

5.2 Results

The completed fault tree global connectivity model can be solved to determine those combinations of network elements whose failures will cause the isolation of one or more end users from the remainder of the network (the network cut sets). The results of the fault tree analysis for the simple example network shown in Figure 2 consist mainly of FDDI ring segmentation cut sets, token ring segmentation cut sets, router and switching hub failures, and failures in Concentrator 3. The ring segmentation cut sets contain combinations of cable, hub, and router failures as well as instances of CAU failure and the failures of the other three concentrators. These cut set results *by themselves* do not contain many real surprises as they are simply the list of ways that the network can become segmented. An experienced network designer would anticipate most of them based on a thorough inspection of the network. This might not be the case, though, if a larger and more complex network were to be used in the analysis. However, the real value of the cut set results is that they form the necessary bridge to the world of importance analysis.

An importance analysis of the cut sets for this example network shows first that there are several single points of failure in the network (routers, the hub, etc.). Intuition tells us that these are important to the reliability of the network. This intuition is confirmed by the Risk Increase importance measure. However, depending on the relative failure rates of *all* of the components, these single failure points may not be the best place to invest money to achieve improved overall system reliability. For example, if FDDI ring 1 is of an unreliable vintage, we may achieve a greater increase in reliability (even possibly at a reduced cost) by upgrading

that equipment even though two failed components are required to segment an FDDI ring. The point is that the importance measures provide us with a quantitative relative ranking of how much reliability improvement can possibly be obtained by improving each network element. This information is key to any effective cost-benefit decision analysis. The information is also in a form that can be readily used to support discrete optimization studies (including genetic algorithms). These powerful tools can be used to help ensure that the maximum network improvement is being attained given the resources available.

6. Extensions of the Method

The method described in the preceding sections provides insights regarding the local and global connectivity conditions. However, simple connectivity is an inadequate measure of network success or failure. Availability of network services and "quality of service issues" are important measures of whether the network is truly successful from the user's perspective. In addition, the restriction of this method to hierarchical networks is a potential liability as the information services world migrates toward the flat network architectures provided by ATM and other technologies. While space constraints prevent us from addressing these issues in detail, we would like to point out some extensions to the plug-and-play method that resolve some of these concerns.

6.1 Network Services

Network services provide individual users with a wide variety of capabilities, including file and printer sharing, mail and hypertext communications, client-server computing environments, and even important parts of basic network communications fabric such as Domain Name Services. Most users consider the availability of at least some of these services to be critical to the success of their daily duties. Therefore, in order to model the reliability of the network from the user's perspective, we must examine the reliability of network services. 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 to model both network connectivity and network services can be constructed based upon the following success criteria: global connectivity must be maintained *and* servers must be available to provide all necessary network services. In a network where a single server provides all network services, the applicable fault tree model simply consists of a logical AND condition of the availability of the server machine and the network connectivity model we developed in previous sections. For more advanced networks with multiple and possibly redundant servers, the single server in the AND 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 would be easy to construct given the network specifications and, while it cannot be developed explicitly as part of the plug-and-play fault tree development methodology, it can be developed automatically using an "interview paradigm" in which the fault tree development software obtains information from the analyst in an automated "question and answer" format.

6.2 Quality of Service Issues

Many historical networking standards have transmitted data on a "best effort" basis. There were no guarantees that data would arrive at its destination "on time," or that a particular data rate would be available. However, advanced networking standards such as ATM allow the network to negotiate quality of service guarantees with the user (maximum delay times, minimum data transfer rates, etc.).[9] The plug-and-play paradigm considers only network connectivity and not the quality of network services. Yet, a simple change will allow this method to model connectivity for guaranteed bit rate services and for guaranteed maximum delay time services (*if* the delay time is specified on a per segment basis within the network and not on an end-to-end communications basis). To construct such a model, one simply builds and solves the global connectivity model as described previously. The network is then examined to identify network elements that are incapable of providing the desired level of service. These elements are identified as "already failed" in the cut sets (failure probability of unity) because they fail to provide the required level of service. Cut set quantification and importance computations are then carried out as before to

determine the reliability of the network under these conditions.

Since the ability of each network element to meet particular levels of service will vary over time based on the exact level of service required as well as existing network traffic patterns, a network designer should examine the fault tree analysis results in light of a number of postulated traffic patterns, congestion conditions, and service requirements to get a good overall picture of the network's expected behavior. This can be done very rapidly since it only requires varying the quantitative input to the fault tree model and not the model structure itself. For this reason, the fault tree model provides a very good tool that network designers can use to rapidly evaluate the response of the network to a wide variety of possible network conditions. One could also consider using this method to help a network manager understand in-progress network service problems. If the network manager could quickly inform the model of the condition of various network elements, the rapid quantification of the cut sets could provide the manager with additional diagnostic information that could be used to better evaluate and respond to the problem at hand.

6.3 Nonhierarchical Network Architectures

The plug-and-play analysis methodology described in this paper is applicable only to networks that are either hierarchical or nearly hierarchical. While this assumption is valid for many current-generation and legacy networks, future networking standards such as ATM will allow for arbitrarily interconnected nonhierarchical ("flat") networks. Direct extension of the plug-and-play methodology to these types of networks would rapidly become unmanageable because of the wide variety of possible paths that data can take to travel between end-user devices. For this reason, Sandia National Laboratories is developing an efficient search algorithm that will work directly from a network architecture diagram to find connectivity cut sets like those produced by the plug-and-play fault tree method. Unlike previous network reliability analysis methods (most of which were based on a path set algorithm),[10],[11] this method finds cut sets directly and efficiently without the need for the computationally intensive mathematical "duality" operation to transform path sets into cut sets. Since this algorithm produces cut sets directly, its results can be used as the basis for an importance analysis using the same methods and tools that were used in the plug-and-play method. The details of this search method will be described in a later paper.

7. 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 communications network architectures. We have demonstrated that many aspects of hierarchical communications networks can be modeled using a plug-and-play fault tree analysis technique. We have demonstrated that the types of results that can be obtained from PLMs can be of great practical value to network designers and analysts. These PLM techniques are not intended 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 optimal network systems.

Acknowledgments

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

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] R.O. Onvural, *Asynchronous Transfer Mode Network Performance Issues*, 2nd Edition, Artech House, 1994.

[10] 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.

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