WSRC-MS-99-00543

DESIGN AND ANALYSIS OF PACKET SWITCHED NETWORKS FOR CONTROL SYSTEMS*

Edward Kamen[1], Payam Torab[2], Kenneth Cooper[3], and George Custodi[4]

This document was prepared in conjunction with work accomplished under Contract No. DE-AC09-96SR18500 with the U. S. Department of Energy.

Abstract

This paper contains a methodology for analyzing and designing a computer network for application to complex control systems. The focus is on the analysis and design of a local area network (LAN) for realizing the high-level control network that interconnects input-output controllers with devices for monitoring and analysis and with high-level controllers such as supervisory PLCs. Part of the development given in this paper can also be applied to the device-level network (fieldbus) that interconnects input-output controllers with sensors,

* This work was supported in part by ERDA under Contract No. 97090.

[1] School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, 30332-0250. e-mail: kamen@ee.gatech.edu

[2] School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, 30332-0250. e-mail: ptorab@ee.gatech.edu

[3] Westinghouse Savannah River Co., Building 730-B, Aiken, SC 29808. e-mail:ken.cooper@srs.gov

[4] Westinghouse Savannah River Co., Building 703-45A, Aiken, SC 29808. e-mail: george.custodi@srs.gov

actuators, and other devices in the system being controlled. The high-level network and the device-level network form a two-layer architecture that is typical in control applications. A procedure is given for generating a network design with a hierarchical hub topology having full redundancy. Then in terms of a graph model of the network, procedures are given for studying network availability and analyzing the information flow rates through the links and internal nodes of the network

1. Introduction

The current state of the art in the design of computer networks for complex, highly distributed control systems is experienced based. The usual approach is to evaluate network performance data from similar type systems and then to purchase the highest performing equipment the project funds will support. This approach frequently leads to expensive systems that fail to meet performance specifications. The work presented in this paper is an attempt to provide an analytic basis for making network topology and component choices. The network issues that initiated this work were network availability and the capability to transfer a required amount of information in a specified time. Given system performance requirements, the approach developed in this paper can be used to modify an existing design so as to meet the requirements. The approach ensures that equipment is added or upgraded only where a performance gain can be realized.

The paper is organized as follows. After a brief introduction to the network graph model, a heuristic approach to generate an initial design is discussed in Section 3. The initial design is a typical two or three-level architecture based on two or three classes of network switch devices. Various degrees of redundancy can be added (heuristically) at this stage to improve the network reliability. Network reliability and availability analysis is discussed in detail in Section 4. Specifically, it is shown how the initial design can be further refined to achieve a specified network availability. Finally, a load analysis technique is developed in Section 5 that enables the designer to compute the maximum offered load to all network devices and decide on the capacity of these devices.

2. Network Graph Model

Given a collision-free packet-switched network, let $G(t) = [X(t), U(t)]$ denote the graph model representing the network interconnection at time t, where $X(t)$ is the set of external nodes (end nodes) and internal nodes at time t and $U(t)$ is the set of links connecting the external and internal nodes at time t. The graph $G(t)$ is referred to as the *operational network* at time t. We further assume that $G(t)$ defines a tree; that is, the operational network at any time t does not contain any cycles. This is the case for switched Ethernet with full duplex links and some ATM network topologies.

For standard non-switched Ethernet and device-level networks, the operating paths between the nodes do not change as a function of time, and thus for such networks $G(t)$ is independent of time. However, in a switched network such as switched Ethernet or ATM, the paths between internal nodes can change as a result of the switching actions. Hence, for these

networks the network graph model is time varying. When $G(t)$ varies as a function of time, the *total network* is given by the *total graph* $G=[X,U]$, where X is the union of the elements of $X(t)$ and U is the union of the elements of $U(t)$ as t ranges over the interval of operation. In general, the total graph will have cycles resulting from the existence of multiple paths between the nodes, and thus it does not define a tree.

3. Network Design

The starting point to network design is the specification of the number of end nodes, that is, the number of nodes to be connected to the network, and the physical location of the end nodes. To carry out network design, it is also necessary to know the expected *information flow rate* from end node to end node for all of the end nodes to be connected to the network. We shall specify this in terms of the *maximum flow rate* a_{ij} from end node i to end node j.

Given the number of end nodes, the location of end nodes, and the maximum flow rates a_{ij}, the problem is to design a network that has a desired reliability and availability and an acceptable performance. Network design can be carried out by first generating an initial design "on paper", and then modifying the design based on an analysis of reliability/availability and the offered loads to links and internal nodes using the procedures given in the following two sections. The generation of an initial design requires that one specify a topology for the network that describes the interconnection of the network components. This is equivalent to specifying the network graph model $G(t)=[X(t),U(t)]$ that was defined in the previous section.

The most common topology in use today (particularly for switched Ethernet) is a *hierarchical hub (collapsed star) configuration*. An example of a two-level hierarchical structure is shown in Fig. 1. As seen from the figure, the top level of the network consists of the backbone and the lower level contains three hubs covering the end nodes. The backbone and the hubs are switches and the links provide full-duplex communication, and thus there are no packet collisions in the network.

The number of levels in a hierarchical hub configuration depends on the number of end nodes, the physical location of the end nodes, and the number of ports on the switching hubs. We shall next give a procedure for generating an initial hierarchical hub design with an

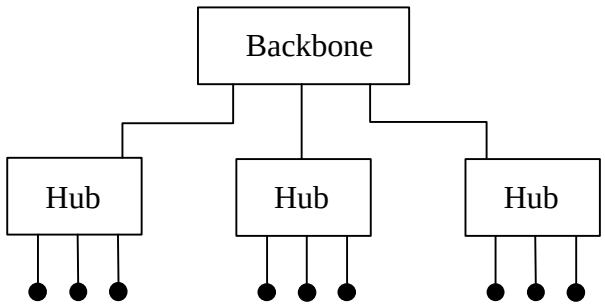

Fig.1 A two-level hierarchical hub configuration

appropriate number of levels. In this procedure it is assumed that the network can be built using high-capacity (backbone) switches, intermediate-capacity switches, and low-capacity switches. The steps of the procedure are as follows:

1. Determine the number N of end nodes, the location of the end nodes, and the maximum flow rates a_{ij} for $i, j = 1, \ldots, N$.
2. Determine the maximum flow rate a_i to and from node i, where a_i is given by
$$a_i = \sum_{j=1}^{N} (a_{ij} + a_{ji})$$
3. Order the end nodes in terms of the flow rate a_i, beginning with the end node having the smallest flow rate.
4. Starting with the end nodes having the smallest flow rate, group the nodes that are in close physical proximity by connecting them to low-capacity switches.
5. Continue grouping the end nodes until the total flow for a group exceeds the capacity of the low-level switch. If the grouping of the nodes does not result in a flow exceeding the capacity of the low-level switch, go to Step 7; otherwise, go to the next step.
6. Group the remaining end nodes that are in close proximity by connecting them to intermediate-capacity switches.
7. Connect the low-capacity switches and the intermediate-capacity switches to a high-capacity switch that will serve as the backbone of the network. This will result in a two-level hierarchy. If the high-capacity switch does not have a sufficient number of ports, connect the low-capacity switches and the intermediate-capacity switches to two or more high-capacity switches, and then connect these switches to a backbone. This will result in a three-level hierarchy although the same capacity switches are used for the backbone (the top level) and the first level below the backbone. Another option is to connect some or all of the low-capacity switches to intermediate-capacity switches, which are then connected to the backbone. This will also result in a three-level hierarchy.

The above procedure generates a network with no redundancy, and thus the network graph model $G = [X, U]$ does not have any cycles, forming a tree. Also note that in this case the graph G is independent of time since there is no switching to alternate paths between nodes. For any design generated via the above procedure, the reliability and availability, and the offered load to all the network links and nodes can be found using the approaches given in Sections 4 and 5. Based on this analysis, the initial design can be modified and then re-analyzed in order to iterate to an acceptable design.

To achieve a desired reliability for a controls application, it is very likely that some degree of redundancy will be necessary; that is, it will be necessary to have alternate paths between some nodes. It is important to emphasize that for Ethernet it is not possible to have more than one operational path between a pair of nodes at the same time. However, by using Ethernet switches that support the Spanning Tree Algorithm and Protocol (IEEE Standard 802.1d), it is possible to provide alternate paths in the network. Using the Spanning Tree Algorithm, the switches learn the multiple paths, and then eliminate them by disabling switch

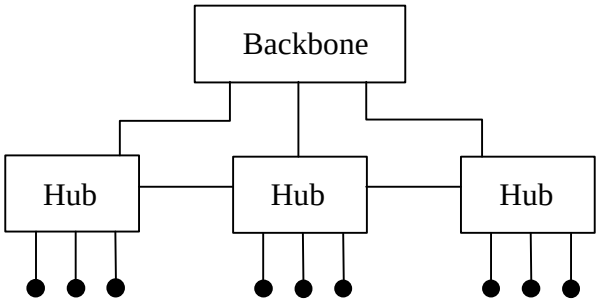

Fig. 2 Two-level hierarchical network with redundancy

ports to form a tree. The tree is called a *spanning tree* since it connects (spans) all the end nodes in the network. When the network software detects a failure, an alternate path is enabled. This corresponds to the network graph model $G(t) = [X(t), U(t)]$ switching from one tree to another tree whenever an alternate path is enabled.

Various degrees of redundancy are possible in the network. In the hierarchical hub configuration discussed above, redundancy can be added by interconnecting switches in the same level. For example, if this is done for the two-level structure in Fig. 1, we obtain the network in Fig. 2. If one or more of the links connecting the hubs to the backbone fail and/or the backbone fails, the redundant connections between the hubs can be enabled so that all the end nodes remain connected.

If any one of the hubs or any one of the links from the end nodes to the hubs fail in the network in Fig. 2, complete connectivity of the network will be lost; that is, various pairs of end nodes will not be able to communicate. To prevent this, we must add additional redundancy so that there is an alternate link from every end node to a separate hub. This can be accomplished by using two-port network interface cards (NICs) on the end nodes.

Suppose that two-port NICs are used on each of the end nodes for the network in Fig. 2, and each end node is connected using two separate links to two different hubs in the network. Then a single failure of any link or internal node in the network will not result in a loss of connectivity; that is, all end nodes will be able to communicate, assuming that end nodes do not fail. This situation is referred to as *full redundancy*, meaning that any single failure of a link or internal node in the network does not affect network connectivity.

For a network given by a hierarchical switching hub configuration, rules can be specified for achieving full redundancy. First, we denote the levels of the hierarchy by Level 1, Level 2, Level 3, and so on. Level 1 is the backbone, Level 2 consists of switches that connect to the backbone, Level 3 consists of switches that connect to the switches in Level 2, and so on. End nodes may be connected to switches at any level in the hierarchy, including the backbone. In terms of this setup, the rules for achieving full redundancy are:

1. Every end node has a two-port NIC with links connected to two different switches in the same level.
2. All switches in Level 2 (below the backbone) are connected together to form a string.

3. Every switch in Level 3 is connected to two separate switches in Level 2, and if there are additional levels, each switch in these levels is connected to two separate switches in the level above.

Additional redundancy can be achieved by connecting the switches in Level 3 together to form a string, and doing the same for any additional levels.

In the configuration that results from applying the above rules, the backbone is not duplicated. If the backbone fails, the string connection of switches in Level 2 takes over. This may be acceptable for temporary operation until the backbone is repaired or replaced. However, since the switches in Level 2 will most likely have much less capacity than the backbone, the string connection of the Level 2 switches may not be able to handle the traffic that would normally go through the backbone. Determining the offered loads on the Level 2 switches in case of a backbone failure can be carried out using the load analysis given in Section 4.

There are possible variations on the configuration that results from applying the above rules. For example, we could add an alternate backbone with every switch in Level 2 connected to both the primary and alternate backbones. In this case, the switches in Level 2 do not have to be connected together in a string. If a port on the primary backbone fails, a port on the alternate backbone can be enabled so that complete connectivity is maintained.

The reliability and availability, and loading for configurations with redundancy can be analyzed using the results in the next two sections. As noted above, an initial design can be modified based on these analysis tools in order to achieve an acceptable design.

4. Network Reliability and Availability

Let X denote the time to failure (TTF) due to hardware or software of a network component (link or node) assuming that the component is new at time $t = 0$. In general, X is a random variable with the probability distribution $F(t)$ defined as $F(t) = P(X \le t) =$ probability that the TTF X is less than or equal to t.

The reliability $R(t)$ of the component is defined by $R(t) = 1 - F(t)$. By definition of $F(t)$, $R(t)$ is equal to the probability that the TTF X is greater than t. An equivalent characterization of the reliability $R(t)$ is that it is the probability that the component is operational over the time interval from 0 to t.

It is known that the general form of the reliability function $R(t)$ is

$$R(t) = \exp\left[-\int_0^t \lambda(t)\, dt \right] \quad , \; t > 0 \qquad (1)$$

where $\lambda(t)$ is the *instantaneous failure rate* defined as follows: Letting Δt denote a small interval of time, $\Delta t\, \lambda(t)$ is equal to the probability that the TTF X is between t and $t + \Delta t$ given that the component did not fail over the time interval from 0 to t. It is assumed that the component has been *burned in*, so that the initial value $\lambda(0)$ of the instantaneous failure rate is equal to zero. This is a reasonable assumption for nodes consisting of electronic devices such as repeaters or switches, since vendors operate these devices through the infant mortality time period before shipment. It is also a reasonable assumption for links since once a link is connected, tested, and proven to be operational, the instantaneous failure rate of the link should initially be zero.

If the component is aging as time goes by, the probability it will fail in the time interval from t to $t + \Delta t$ given that it did not fail over the time interval from 0 to t will increase as t increases. Thus for components with aging, $\lambda(t)$ will increase as t increases. Aging can be taken into account by using the Weibull distribution to model the time to failure X. For this distribution, the reliability function of the component is given by

$$R(t) = \exp\left[-\left(\frac{t}{\theta}\right)^m\right] \quad , t > 0 \tag{2}$$

where θ and m are positive real numbers with $m > 1$. Writing $R(t)$ in the form (1), we see that the instantaneous failure rate is

$$\lambda(t) = \frac{m}{\theta}\left(\frac{t}{\theta}\right)^{m-1} \tag{3}$$

Note that when $m = 1$, $\lambda(t)$ is a constant equal to $1/\theta$ and thus in this case there is no aging. For $m > 1$, $\lambda(t)$ clearly increases as t increases and $\lambda(0) = 0$, and thus there is aging with the initial value of the instantaneous failure rate equal to zero. The larger the value of $m > 1$, the faster the component ages. For a network component, a value of m equal to 2 or 3 is most likely sufficient to capture the aging of the component.

The *mean time to failure* (MTTF) is the expected value $E[X]$ of the time to failure X. For the Weibull distribution with $R(t)$ given by (2), the MTTF is given by

$$\text{MTTF} = \theta\, \Gamma(1 + \frac{1}{m}) \tag{4}$$

where $\Gamma(x)$ is the gamma function defined by

$$\Gamma(x) = \int_0^\infty \tau^x \exp(-\tau) d\tau$$

When the MTTF of the network component is specified, the parameter θ in the Weibull distribution can be determined using (4).

The analysis of network availability given below is developed in terms of the MTTF of the network components, and thus the MTTF of the components must be specified. The MTTF for various network devices such as switches can be obtained from vendors. For a link, the most likely cause of failure is the connection of the link at the end points. It is also possible that a link may fail at some point along its length, due for example to the cable being cut. These factors play a major role in determining the MTTF for a link. It may be possible to generate a good estimate of the MTTF of a link by using link failure information for some existing network. In the analysis given below, we take the MTTF of a link to be 50 years.

The availability $A(t)$ of a network component at time t is defined to be the probability that the component is operational at time t. In the following development we assume that when a component fails it is replaced or repaired to new condition. As a consequence of this assumption, the mean time between failure (MTBF) is equal to the mean time to failure (MTTF) defined above. Hence, the MTBF is also given by (4) when the TTF X is modeled by a Weibull distribution.

A differential equation for $A(t)$ can be derived as follows: With Δt equal to a small interval of time, by the total probability theorem the availability $A(t+\Delta t)$ at time $t+\Delta t$ can be expressed in the form

$$A(t+\Delta t) = [1-\Delta t\,\gamma(t)]\,A(t) + \Delta t\,\nu(t)[1-A(t)] \qquad (5)$$

where $1-\Delta t\,\gamma(t)$ is the probability the component is operating at time $t+\Delta t$ given that it is working at time t and $\Delta t\,\nu(t)$ is the probability that the component is working at time $t+\Delta t$ given that it is not working at time t. Thus, $\Delta t\,\gamma(t)$ is the probability that the component is not working at time $t+\Delta t$ given that it is working at time t, and $\nu(t)$ is the instantaneous repair rate of the component.

Rewriting (5) gives

$$\frac{A(t+\Delta t)-A(t)}{\Delta t} = -[\gamma(t)+\nu(t)]A(t)+\nu(t)$$

and in the limit as $\Delta t \to 0$, we have the differential equation

$$\frac{dA(t)}{dt} = -[\gamma(t)+\nu(t)]A(t)+\nu(t) \qquad (6)$$

Assuming the component is new at time $t=0$, the initial condition for (6) is $A(0)=1$. The availability $A(t)$ can then be determined by solving (6) with $A(0)=1$.

In the case when $\gamma(t)$ and $\nu(t)$ in (6) are constants so that $\gamma(t)=\gamma$ and $\nu(t)=\nu$ for all t, $1/\gamma$ is the mean time between failure (MTBF) and $1/\nu$ is the mean time to repair

(MTTR). In this case, the solution $A(t)$ to (6) decreases as t increases, and in the limit as $t \to \infty$, converges to the value

$$A(\infty) = \frac{\text{MTBF}}{\text{MTBF} + \text{MTTR}} \tag{7}$$

The value $A(\infty)$ given by (7) is the steady-state value of the availability. From the form of (7) we see that $A(\infty)$ is the fraction of time that the component is operational in steady-state operation.

For components with aging, $\gamma(t)$ will vary as a function of t, and thus the assumption that it is constant is not valid. In this case, the determination of $\gamma(t)$ is very difficult since the probability the component is not working at time $t + \Delta t$ given that it is working at time t depends on when the component was last repaired. However, an approximate solution can be generated by assuming that the component was replaced or repaired to new condition at time $t - \text{MTBF}$. Then $\gamma(t) = \lambda(\text{MTBF} + \Delta t)$, where $\lambda(t)$ is given by (3). Inserting $t = \text{MTBR} + \Delta t$ into (3) and using the fact that Δt is a small interval gives

$$\gamma(t) = \frac{m(\text{MTBF})^{m-1}}{\theta^m} \tag{8}$$

Then since MTBF=MTTF and MTTF is given by (4), using (4) in (8) gives

$$\gamma(t) = \frac{m\left[\Gamma\left(1 + \frac{1}{m}\right)\right]^m}{\text{MTBF}}$$

If $\nu(t)$ is a constant equal to $1/\text{MTTR}$, it then follows that as $t \to \infty$, the solution $A(t)$ to (6) converges to

$$A(\infty) = \frac{\text{MTBF}}{\text{MTBF} + (\text{MTTR})C} \tag{9}$$

where

$$C = m\left[\Gamma\left(1 + \frac{1}{m}\right)\right]^{\text{m}} \tag{10}$$

Some values for C are shown in Table 1.

Table 1 Some values of C for different values of m

m	1 (No aging)	2	3	4
C	1.000	1.570	2.136	2.700

From (9) we see that a larger $m > 1$ (faster aging rate) results in a larger C and therefore a smaller steady-state availability. Note however that the effect in going from $m = 1$ to $m = 4$ is very small since the change in C is not that great and the MTBF for a network component will be much larger than the MTTR.

Now suppose that a network contains Q components with $R_i(t)$ and $A_i(\infty)$ equal to the reliability function and steady-state availability of component i in the network. The reliability function $R(t)$ of the network is the probability that all end nodes can communicate over the time interval from 0 to t assuming that the network is in new condition at time 0. The availability $A(t)$ of the network is the probability that all end nodes can communicate at time t. The steady-state availability $A(\infty)$ is equal to $A(t)$ in the limit as $t \to \infty$.

As discussed in the previous section, if the network has no redundancy so that the network graph model has no cycles, then the network is a tree. In this case, the network reliability $R(t)$ is equal to the product of the component reliabilities $R_i(t)$ and the steady-state network availability $A(\infty)$ is the product of the component steady-state availabilities $A_i(\infty)$; that is

$$R(t) = R_1(t)R_2(t)\ldots R_Q(t) \quad \text{and} \quad A(\infty) = A_1(\infty)A_2(\infty)\ \ldots\ A_Q(\infty) \qquad (11)$$

For an example of the availability computation, suppose that the network consists of 500 links and 25 internal nodes (switches, for example) with the MTTF of a link equal to 50 years and the MTTF of a node equal to 20 years. This number of links and nodes, and the values for the MTTF, may be fairly realistic for the high-level network of a complex control system. Assuming that the MTTR of a link or node is given in hours and $C = 1$, the steady-state availability of the overall network is given by

$$A = \left[\frac{(50)(365)(24)}{(50)(365)(24)+Y}\right]^{500}\left[\frac{(20)(365)(24)}{(20)(365)(24)+Y}\right]^{25} \qquad (12)$$

The network availability for some MTTR values is shown in Table 2.

Table 2 Steady-state availability for different values of MTTR

MTTR (Hours)	1	2	4	8
Availability	0.9987166	0.9974348	0.9948762	0.9897787

Using the above values for A, we can compute the mean time between failures in the overall network: Letting D denote the mean time between failures in the overall network, we have that $A = D/(D + \text{MTTR})$ and thus, $D = (A)(\text{MTTR})/(1 - A)$. Inserting the values for A and MTTR given above results in $D \approx 32$ days for all four values of MTTR. Thus, in steady-state operation a failure can be expected every 32 days on the average. Of course, the extent of the loss of network connectivity resulting from a failure depends on what failed. This is a different problem that is not considered in this work.

The availability obtained above is not likely to be acceptable for a control application, and thus there is a need for redundancy. Hence, suppose that redundant links and internal nodes are added to the network and that these components are switched in via the software when needed. Given the nature of redundancy in a network, the redundant components will be operational even when they are not used, and thus the redundant components are *active standbys*. The steady-state availability of a network with redundancy can be computed by determining all the *spanning trees* in the network, where a spanning tree is a *minimal* tree that contains all of the end nodes. By minimal, we mean that there is no subgraph of the tree that also spans all the end nodes. Given the network graph model $G = [X, U]$, the spanning trees can be determined using existing algorithms [GoM84] and thus they are readily computed.

For an example, suppose that original nonredundant network is a two-level hierarchical configuration with a total of 476 end nodes, 500 links and 25 internal nodes. Since one of the 25 internal nodes is the backbone, the remaining 24 internal nodes are the hubs in the lower level. Of the 500 links, 476 are links between the lower-level hubs and the end nodes, and the remaining 24 are the links between the lower-level hubs and the backbone. Assuming the MTBF of a link is 50 years, the MTBF of an internal node is 20 years, and the MTTR of a node or link is 4 hours, as computed above the availability of the overall network is equal to 0.9948762. Now suppose that using two-port NICs, each end node is connected to two separate hubs in the lower level. This will require that the number of lower-level hubs is increased from 24 to 48 and the number of links is increased from 500 to 1000. The increase in the number of links includes the additional links resulting from the connection of each additional hub in the lower level to the backbone. We assume that the hubs in the lower level are not connected to each other.

For this configuration, it can be shown that the overall network availability A is given by

A = (availability of backbone)×(product of the availabilities of the links connecting the hubs to the backbone)×(product of the availabilities of the links with active standbys connecting the hubs to the end nodes)×(product of the availabilities of the hubs with active standbys)

The availability of a link with an active standby is equal to $2A_L - A_L^2$, where A_L is the availability of a link and its standby. The availability of a hub with an active standby is

equal to $2A_H - A_H^2$, where A_H is the availability of a hub and its standby. Therefore, the network availability is

$$A = A_B A_L^{48} \left[2A_L - A_L^2\right]^{500} \left[2A_H - A_H^2\right]^{24} \qquad (13)$$

Evaluating (13) with

$$A_L = \frac{(50)(365)(24)}{(50)(365)(24)+4},\ A_H = A_B = \frac{(20)(365)(24)}{(20)(365)(24)+4}$$

results in $A = 0.9995389$. This is a major improvement in availability; in fact, the mean time between failures of the redundant network is approximately 361 days, whereas, for the nonredundant network the MTBF is 32 days.

The availability can be further enhanced by adding an alternate backbone with each lower-level hub connected to both the primary and alternate backbones. In this case, it can be shown that the availability is $A = 0.9999999412662$. With this availability, the mean time between failures is 7,774 years, which means that there will be no failures that result in a loss of network connectivity during the lifetime of the network! Components will fail, but when this happens the standby units take over so there is no loss of connectivity. This result on the availability assumes that any primary or standby unit that fails during the lifetime of the network is repaired within a mean time of 4 hours of the failure.

5. Network Load Analysis

As we saw earlier, the topology of switched networks can be conveniently described by a graph model. Graph models have been used in a few studies like [GuR91] and [Gur92] for performance modeling of local-area and heterogeneous data networks, and more research in this area is expected to emerge in the coming years.

The concept of *network flow* has been used to model many problems in transportation and communication networks ([GoM84], [Ker93]). In computer communication networks, flows can represent either the total amount of information transferred between two nodes or the amount of information per unit time. The best analogy is the use of flows in electric circuits to represent the flowing electric charges or currents. Here, we take the *rate-based* approach, and use flows to describe the *rate* of information transferred over the network. The notion of flow used here is inspired by the similar idea of a *communication session* in [Cru91a] and [Cru91b].

Let T_{ij} denote the information flow in packets per second between end nodes i and j of the network. We define the *traffic matrix* of the network to be the $n \times n$ matrix $\mathbf{T} = (T_{ij})$, where n is the number of end nodes. The traffic matrix $\mathbf{T}$ may consist of the maximum flows, average flows, etc. of the information flow between the end nodes. In an actual network, the diagonal elements of the traffic matrix are all zero, as no end node generates data packets destined for itself. However, as we shall see shortly, a basic step in the load analysis of tree

networks is to remove some end nodes and continue the analysis on a *trimmed* network. The resulting network will be an *artificial* network with end nodes that may correspond to internal nodes of the original network. To keep track of the flows inside the network as it gets smaller, it is convenient to assign some *internal traffic* to the end nodes that actually represents part of the total flow through those nodes in the bigger network. The traffic matrix is therefore assumed to have nonzero diagonal elements in general. In the following analysis, it is assumed that all network traffic is generated by and destined for the end nodes.

Given the traffic matrix **T** and the network topology specified by a directed graph $G = [X,U]$, we next develop an algorithm for determining the *total flows through links and nodes* of a network. Intuitively speaking, the total flow through a link (node), also called the *offered load* to the link (node), is the amount of information passing through the link (node) per unit time. The *load analysis problem* for a graph can be described as computing the offered load to all links and nodes of the graph. For a complete discussion of the subject see [KaT98].

A directed graph has directed links, in the sense that each link is an *ordered* pair of nodes (i, j), where i is called the *source* node and j is called the *sink* node. Node j is a *successor* of i, and i a *predecessor* of j, if there exists a link (i, j) in the graph. The set of successors of i defines a mapping denoted by Γ_i. We further assume that $G = [X,U]$ is an *arborescence*; that is, $G = [X,U]$ is a directed graph with a root $r \in X$ such that for every node $j \in X$, there is a path from r to j in G. The d*epth* of a node belonging to an arborescence is the length of the longest path starting from that node. The *depth* of an arborescence is the maximum depth of its nodes**.** For example, the graph shown in Fig. 3 is an arborescence with root 7, because there is a path from node 7 to every other node in the tree. The arborescence has a depth of two, which is also the depth of its root node. Note that the notion of depth is not the same as the notion of level considered in Section 3.

In what follows, we denote the set of end nodes by X_t. For any set $Y = \{y_1,...,y_m\} \subseteq X_t$, $\mathbf{e}_{n,Y}$ denotes an n-vector with ones at positions $y_1,...,y_m$ and zeros elsewhere. When there is no ambiguity about the size of the vector we will use the simpler notation $\mathbf{e}_Y$. If Y has a single member y, we use the simplified notation $\mathbf{e}_{n,y}$ to represent the vector. The proofs of the following results are not included due to space limitations. The proofs can be found in [KaT98].

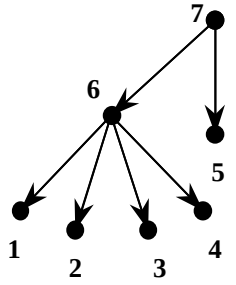

Fig. 3. An arborescence having depth 2.

The following theorem is the first step toward the load analysis of tree networks. It basically shows that the total flow through *all nodes with a depth of one* can be found by simple matrix operations on the traffic matrix of the network.

Theorem 1 The total flow through any node i with a depth of one is given by

$$\lambda_i = \mathbf{e}_{\Gamma_i}^T \hat{\mathbf{T}} \mathbf{e}_{X_t - \Gamma_i} + \mathbf{e}_{X_t - \Gamma_i}^T \hat{\mathbf{T}} \mathbf{e}_{\Gamma_i} + \mathbf{e}_{\Gamma_i}^T \hat{\mathbf{T}} \mathbf{e}_{\Gamma_i} \tag{14}$$

where $\hat{\mathbf{T}}$ is the *apparent traffic matrix* defined as

$$\hat{T}_{jk} = \begin{cases} T_{jk} \; ; \; j \neq k \\ T_{jk} \; ; \; j = k \text{ and the predecessor for end node } j \text{ has a depth greater than one} \\ 0 \quad ; \; j = k \text{ and the predecessor for end node } j \text{ has a depth of one} \end{cases} \tag{15}$$

Using Theorem 1, one can find the total flow through all nodes of a tree network that have a depth of one. The next theorem paves the way for a complete analysis of the network by showing that the graph model of the network can be replaced by a simpler one after all nodes with a depth of one have been covered. It then follows that a *recursive* application of Theorem 1 and the simplification scheme, developed next, can be used to cover all nodes of the network and complete the load analysis.

Theorem 2 Consider a network represented by a tree T and a traffic matrix $\mathbf{T}$. T has n end nodes and a depth of d. Let $V = \{v_1, \ldots, v_q\}$ be the set of depth-1 nodes of T. Let $X_t^* = \bigcup_{i=1}^{q} \Gamma_{v_i}$ be the set of T end nodes with a predecessor in V and assume $\left| X_t^* \right| = p \leq n$. Now consider T', a *trimmed* version of T, obtained by removing all end nodes in X_t^* and links attached to them. The following properties hold for T':

i. T' has a depth of $d-1$.

ii. T' has m end nodes with $m = n - p + q$.

iii. Renumber the end nodes in T' from 1 to m. Define an $n \times m$ *succession matrix* $\mathbf{E}$ as follows. $E_{ij} = 1$ if end node i in T is a successor or the same as end node j in T', and $E_{ij} = 0$ otherwise. Then the total flow through end node j of T' is given by the sum of the elements in row j and column j of the matrix product

$$\mathbf{T}' = \mathbf{E}^T \hat{\mathbf{T}} \mathbf{E} \tag{16}$$

where $\hat{\mathbf{T}}$ is the apparent traffic matrix defined in Theorem 1. *Furthermore, if the above traffic matrix is associated with T', the total flow through all common links and nodes of T and T' will be the same.*

The above theorem is the key to load analysis of switched networks with an arborescence topology. Starting with the graph model of the original network, the theorem can be repetitively applied to the model until it is reduced to a single node, which happens to be the root node for the original arborescence. Part (i) of the theorem guarantees that for an arborescence of depth d, this can be achieved in exactly d steps. At step i (numbered backward from d to 1), the total flows through the end nodes of the network T_i are found by a simple operation on the traffic matrix $\mathbf{T}_i$. Using the theorem, a reduced network T_{i-1} with a depth of $i-1$ and the corresponding traffic matrix $\mathbf{T}_{i-1}$ are generated next, and the process is repeated until the network is reduced to a single node with a depth of zero, completing the analysis. Also, note that because of the tree structure of the network, at each step only one link is connected to each end node of T_i and therefore the total flow through this link is also equal to the total flow through the terminating end node. More precisely, we have the following algorithm based on Theorem 2.

Theorem 3 (Load Analysis of Networks with Arborescence Topology) Consider a network with the graph model T and the traffic matrix $\mathbf{T}$. T defines an arborescence with a depth of d. Set $T_d \coloneqq T$, $\mathbf{T}_d \coloneqq \mathbf{T}$ and $i \coloneqq d$.

1. Set m_i equal to the order of $\mathbf{T}_i$ (number of end nodes in T_i).
2. Do the following steps for $j = 1, \ldots, m_i$:
   - Compute σ_j, sum of the elements in row j and column j of $\mathbf{T}_i$.
   - Find the node v in the original network T that corresponds to end node j in T_i, and set $\lambda_v \coloneqq \sigma_j$.
   - Find the link u in the original network T that corresponds to the link terminating at end node j in T_i, and set $\psi_u \coloneqq \sigma_j$.
3. Construct the reduced graph model T_{i-1} by removing all end nodes of T_i with a predecessor having a depth of one, and the links terminating at those end nodes. Identify and number the new and possibly remaining old end nodes.
4. Find the succession matrix $\mathbf{E}_i$ and the apparent traffic matrix $\hat{\mathbf{T}}_i$ as described in Theorem 2 and compute the traffic matrix for the reduced network $\mathbf{T}_{i-1} \coloneqq \mathbf{E}_i^T \hat{\mathbf{T}}_i \mathbf{E}_i$.
5. Set $i \coloneqq i-1$.
6. If $i > 0$ go to step 1 and stop otherwise. At this point the total flows through all links and nodes of the network have been computed and the load analysis is complete.

Illustrative Example

Consider a switched Ethernet network with 16 end nodes as shown in Fig. 4. Node 16 represents the Master Control Room (MCR) and a major portion of the network traffic is

directed toward this node. The traffic matrix for the *average* flows, expressed in thousands of packets per second for example, is given by

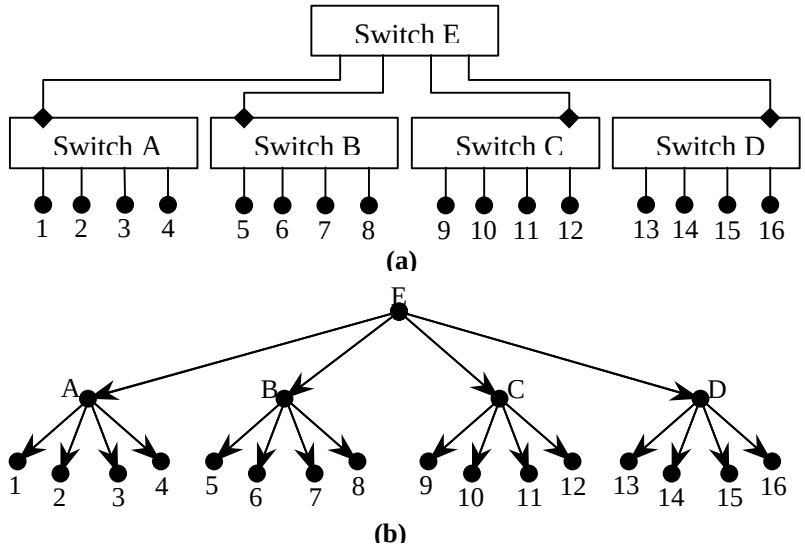

Fig. 4 A switched Ethernet network with 16 end nodes: (a) Network configuration (b) Network graph model

$$\mathbf{T} = \begin{pmatrix}
 & 1.4 & 1.3 & 1.2 & 1.1 & 1.0 & 0.9 & 0.8 & 0.7 & 0.6 & 0.5 & 0.4 & 0.3 & 0.2 & 0.1 & 5.0 \\
1.5 & & 1.4 & 1.3 & 1.2 & 1.1 & 1.0 & 0.9 & 0.8 & 0.7 & 0.6 & 0.5 & 0.4 & 0.3 & 0.2 & 5.0 \\
1.6 & 1.5 & & 1.4 & 1.3 & 1.2 & 1.1 & 1.0 & 0.9 & 0.8 & 0.7 & 0.6 & 0.5 & 0.4 & 0.3 & 5.0 \\
1.7 & 1.6 & 1.5 & & 1.4 & 1.3 & 1.2 & 1.1 & 1.0 & 0.9 & 0.8 & 0.7 & 0.6 & 0.5 & 0.4 & 5.0 \\
1.8 & 1.7 & 1.6 & 1.5 & & 1.4 & 1.3 & 1.2 & 1.1 & 1.0 & 0.9 & 0.8 & 0.7 & 0.6 & 0.5 & 5.0 \\
1.9 & 1.8 & 1.7 & 1.6 & 1.5 & & 1.4 & 1.3 & 1.2 & 1.1 & 1.0 & 0.9 & 0.8 & 0.7 & 0.6 & 5.0 \\
2.0 & 1.9 & 1.8 & 1.7 & 1.6 & 1.5 & & 1.4 & 1.3 & 1.2 & 1.1 & 1.0 & 0.9 & 0.8 & 0.7 & 5.0 \\
2.1 & 2.0 & 1.9 & 1.8 & 1.7 & 1.6 & 1.5 & & 1.4 & 1.3 & 1.2 & 1.1 & 1.0 & 0.9 & 0.8 & 5.0 \\
2.2 & 2.1 & 2.0 & 1.9 & 1.8 & 1.7 & 1.6 & 1.5 & & 1.4 & 1.3 & 1.2 & 1.1 & 1.0 & 0.9 & 5.0 \\
2.3 & 2.2 & 2.1 & 2.0 & 1.9 & 1.8 & 1.7 & 1.6 & 1.5 & & 1.4 & 1.3 & 1.2 & 1.1 & 1.0 & 5.0 \\
2.4 & 2.3 & 2.2 & 2.1 & 2.0 & 1.9 & 1.8 & 1.7 & 1.6 & 1.5 & & 1.4 & 1.3 & 1.2 & 1.1 & 5.0 \\
2.5 & 2.4 & 2.3 & 2.2 & 2.1 & 2.0 & 1.9 & 1.8 & 1.7 & 1.6 & 1.5 & & 1.4 & 1.3 & 1.2 & 5.0 \\
2.6 & 2.5 & 2.4 & 2.3 & 2.2 & 2.1 & 2.0 & 1.9 & 1.8 & 1.7 & 1.6 & 1.5 & & 1.4 & 1.3 & 5.0 \\
2.7 & 2.6 & 2.5 & 2.4 & 2.3 & 2.2 & 2.1 & 2.0 & 1.9 & 1.8 & 1.7 & 1.6 & 1.5 & & 1.4 & 5.0 \\
2.8 & 2.7 & 2.6 & 2.5 & 2.4 & 2.3 & 2.2 & 2.1 & 2.0 & 1.9 & 1.8 & 1.7 & 1.6 & 1.5 & & 5.0 \\
5.0 & 5.0 & 5.0 & 5.0 & 5.0 & 5.0 & 5.0 & 5.0 & 5.0 & 5.0 & 5.0 & 5.0 & 5.0 & 5.0 & 5.0 &
\end{pmatrix}$$

The load on external node i is simply found by summing the elements in row i and column i of the above matrix. These loads are listed in Table 3. As the table shows, the offered load to all external nodes of the network except the master control room is the same.

Table 3 Average offered load to external nodes of the networks in Fig. 4

End Node	1	2	3	4	5	6	7	8
Offered Load	50.6	50.6	50.6	50.6	50.6	50.6	50.6	50.6
End Node	9	10	11	12	13	14	15	16
Offered Load	50.6	50.6	50.6	50.6	50.6	50.6	50.6	150.0

Only the total flow through internal nodes of the networks are computed, as the loads on network switches are the primary parameters of interest in this example. The hierarchical network has a depth of two and therefore all the nodes are covered in two steps. These steps are shown in Fig. 5. The offered loads to switches A, B, C and D are found after the first iteration, as all four switches have the same depth of one. The last node covered is switch E, the root node of the network.

The average offered load to all switch devices is shown in Fig. 6. As seen in the figure, the *backbone* switch E experiences the highest load in the network. One interesting observation about the hierarchical design is that it can be implemented by two distinct classes of switch devices. The *low-level* or *local* switches A, B, C and D undergo a local traffic and have roughly the same amount of offered load, making it possible to select these devices from a class of modest switches without degrading the performance. The *high-level* or *backbone* switch E maintains the connectivity of the entire network and clearly experiences a higher load, suggesting the fact that it should be selected from a class of high-performance switch devices. To ensure that no packets are dropped in internal nodes, the maximum offered load to all these nodes can be computed based on the network maximum flows and the analysis technique developed here. The capacity of each network component is then selected to be higher than its maximum offered load, a design step referred to as *capacity planning*.

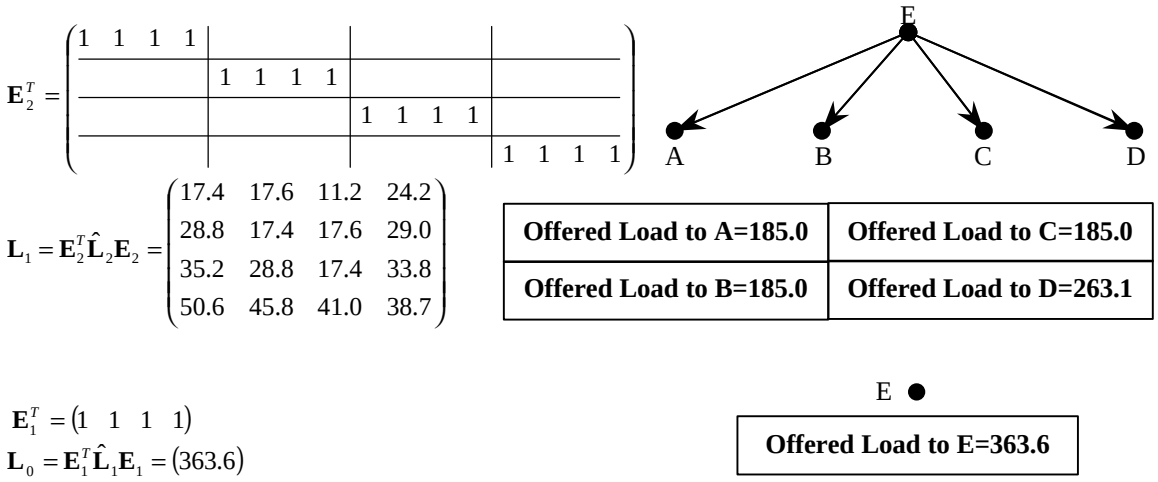

Fig. 5 Load analysis of the network in Fig. 4

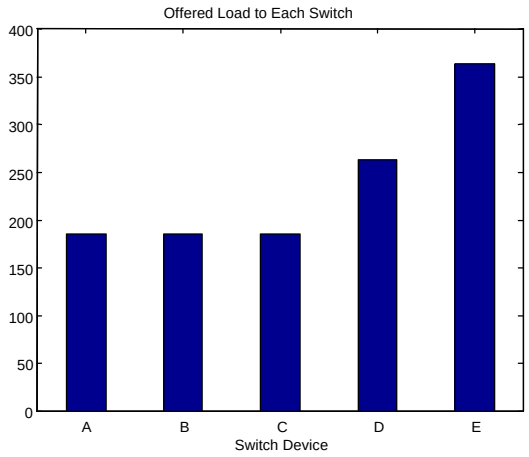

Fig. 6 Average switch loads for the network in Fig. 4

An approximate delay analysis can be done by computing the delay time for each switch device as a function of its offered load, and using the graph model of the network to combine these delay times and find the end-to-end delay times. As a beginning step, we have modeled each switch with a simple M/D/1 queueing system. This means that the packet interarrival times are assumed to have an exponential distribution and the packet service time is the constant transmission time T. The average queueing delay for a switch with offered load λ is given by

$$\overline{W} = \frac{T}{2} \cdot \frac{\lambda T}{1-\lambda T} \quad , \lambda < \lambda_{\max} = \frac{1}{T} \tag{17}$$

Assuming a store-and-forward operation for the switches, the delay time for communication between two end nodes is found by adding the queueing times for the switches on the connecting path and the appropriate number of transmission times depending on the number of switches a packet has to visit. The end-to-end delay times for the hierarchical design is shown in Fig. 7. The packet transmission time T in this analysis has been set to the transmission time for a Fast-Ethernet maximum-sized packet:

$$T = \frac{1518 * 8 \quad \text{bits}}{100 * 10^6 \quad \text{bits/sec.}} = 1.2144 * 10^{-4} \quad \text{sec.} \tag{18}$$

6. Conclusions

This paper is an attempt to develop a systematic design methodology for switched networks in control systems. The starting point is an initial heuristic design based on experience and a knowledge of the performance of similar existing networks. We have shown how this initial design can be further refined to achieve certain reliability and performance requirements. Even with this approach, the network design for large, complex, highly distributed control systems will be an iterative process. The advantage to this approach is that the iterations do not involve hardware and construction costs. Therefore, this approach should lead to network designs that more closely meet performance requirements at a lower cost.

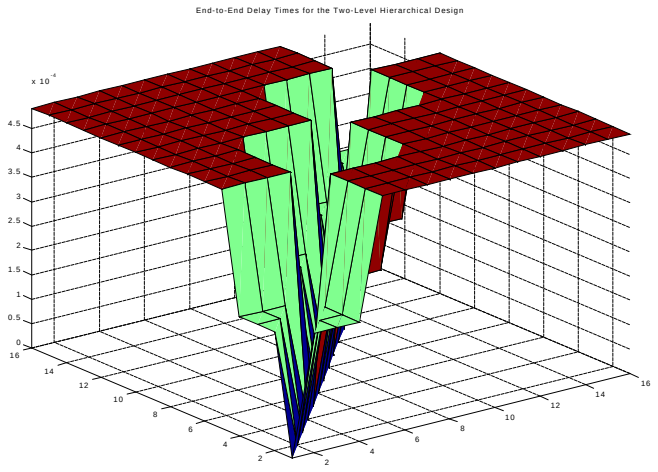

Fig. 7 End-to-end delay time for the two-level hierarchical design

References

[Cru91a] R. L. Cruz, "A Calculus for Network Delay, Part I: Network Elements in Isolation", *IEEE Trans. Inform. Theory*, vol. 37, pp. 114-131, Jan. 1991.

[Cru91b] R. L. Cruz, "A Calculus for Network Delay, Part II: Network Analysis", *IEEE Trans. Inform. Theory*, vol. 37, pp. 132-141, Jan. 1991.

[GoM84] M. Gondran and M. Minoux, *Graphs and Algorithms*, New York: Wiley, 1984.

[GuR91] S. Gupta and K. W. Ross, "Performance Modeling and Optimization of Networks of Bridged LANs", *Queueing Systems*, vol. 9, pp. 113-132, 1991.

[GuR92] S. Gupta and K. W. Ross, "Performance Modeling of Heterogeneous Data Networks", *Annals of Operations Research*, vol. 35, pp. 125-151, 1992.

[KaT98] E. W. Kamen and P. Torab, "Process Control and Information System Network Architecture for Application to Particle Accelerator Facilities", School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, Dec. 1998.

[Ker93] A. Kershenbaum, *Telecommunications Network Design Algorithms*, New York, NY: McGraw-Hill, 1993.