Realizing Fault-Tolerant Interconnection Networks via Chaining

NIAN-FENG TZENG, PEN-CHUNG YEW, AND CHUAN-QI ZHU

Abstract—**A scheme applicable to a wide class of multistage interconnection networks to enhance their fault-tolerant capability is proposed. Multiple paths between each input-output pair of a network are created by connecting together switching elements within the same stage. This scheme provides a network with alternative paths at every stage, requires a simple self-routing algorithm, and allows a network to become more robust as its size increases. An analysis is performed to obtain a quantitative measurement on the reliability improvement of the scheme.**

Index Terms—**Multiprocessors, multistage interconnection networks, network fault-tolerance, reliability analysis, routing schemes.**

I. Introduction

A multiprocessor system consists of many processors and memory modules interconnected by a network. To design a suitable interconnection network has long been recognized as one of the key issues in developing a multiprocessor system because overall system performance relies heavily upon the employed interconnection network. Many multistage interconnection networks (MIN's) have been proposed previously (see the references in [1]).

A MIN has several stages of small switching elements (SE's), which basically are crossbar switches. A path from an input to an output can be established using a destination-tag routing algorithm proposed by Lawrie [2]. However, most of the MIN's have only one unique path between each network input-output pair. Any single failure at a switching element or a link may render some outputs unreachable from certain inputs.

To overcome this problem, many schemes have been introduced to improve network fault-tolerant capability [4]–[11]. These schemes require some kind of redundancy to be built into a network, where redundancy could be in the form of hardware, time [4], or information [7]. Among these schemes, incorporating redundant hardware to provide multiple paths between every input-output pair is most widely used in designing fault-tolerant MIN's [5], [6], [8]–[11].

The interconnection networks proposed in [6], [8], and [9] have alternative paths only at some stages. Should a request in such a network encounter a fault on the way to its destination, it has to backtrack to a stage where alternative paths exist and use an alternative path. The time overhead in backtracking is significant. In addition, the control mechanism is very complicated because the network has to handle bidirectional signal flow. It is possible to eliminate the backtracking overhead by maintaining the status tables of paths. However, extra hardware for maintaining status tables may be substantial, and a new tag may be needed to use an alternative path.

Fault-tolerant MIN's proposed in [5], [10], and [11] provide alternative paths at every stage. A request in such a network can find an alternative path at the same stage where it encounters a fault. A network with this property has *strong reroutability* and is called a *strong reroutable* network. The overall control mechanism is simpler since no backtracking requests or status tables would be required. Although it may require some extra hardware, a strong reroutable network is perferable because the rerouting process incurs much less overhead.

Manuscript received May 26, 1987; revised November 20, 1987. This work was supported in part by National Science Foundation under Grants US NSF DCR84-06916 and US NSF DCR84-10110, the US Department of Energy under Grant US DOE-DE-FG02-85ER25001, and the IBM Donation.

N.-F. Tzeng is with the Center for Advanced Computer Studies, University of Southwestern Louisiana, Lafayette, LA 70504.

P.-C. Yew and C. -Q. Zhu are with Center for Supercomputing Research and Development, University of Illinois at Urbana-Champaign, Urbana, IL 61801.

IEEE Log Number 8719333.

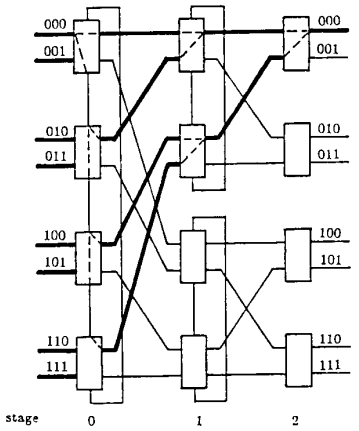

Fig. 1. One way to chain SE's in each stage to provide multiple paths.

In this correspondence, a fault-tolerance scheme which can be applied to a large class of MIN's is proposed. A network constructed with this scheme is a strong reroutable network. It requires a simple routing procedure. In Section II, a fault-tolerant network implemented with the proposed scheme is described. The reliability analysis of the network is presented in Section III. In Section IV, this fault-tolerance scheme is compared to other schemes.

II. Fault-Tolerant Multistage Interconnection Networks

A baseline network [3] built from 2 × 2 switching elements will be used as an example to illustrate our fault-tolerance scheme.

A. Network Configuration

A baseline network of size 8 consists of three stages each of which has four switching elements. A close inspection of a baseline network (see Fig. 1) reveals that a tree can be formed from each output (as the root of the tree) to all of the inputs of the network (as the leaves of the tree) with SE's as its nodes. All of the nodes with the same distance to the root form a level, and they happen to be the SE's in the same stage of the network. Our scheme is simply to chain together all of the nodes (i.e., SE's) in the *same level* of the tree by using extra links between the nodes and, hence, allows each node to have more than one path to the root. Redundant paths can be provided easily this way. Fig. 1 shows one way to connect the SE's in each stage to create redundant paths. The dotted lines in the figure illustrate four possible redundant paths between an input-output pair.

To permit SE's to be chained together, we provide each switching element with a chain-in link and a chain-out link (in addition to the original input links and output links). The "augmented" switching element functions like a 3 × 3 crossbar switch with a modified destination-tag routing algorithm which will be described later.

We use a naming mechanism similar to [3] to describe the configuration of the network. The stages are labeled in a sequence from 0 to $\log_2 N - 1$ with 0 for the leftmost stage. In each stage, an SE is named by the binary representation of its location in the stage, $p_0 p_1 \cdots p_{t-1} (t = \log_2 N - 1)$. Each input/output link of an SE is named by $t + 1$ bits, $p_0 p_1 \cdots p_{t-1} p_t$, in which the leftmost t bits are the same as the binary representation of the SE; the last bit p_t is 0 if the link is the upper link and p_t is 1 if it is the lower link.

Definition 1: A set of SE's in stage i, $0 < i < \log_2 N$, belongs to the same *partition* if $p_0 p_1 \cdots p_{i-1}$ in the binary representation of their names, $p_0 p_1 \cdots p_{t-1}$, has the same value. All of the SE's in stage 0 constitute a partition.

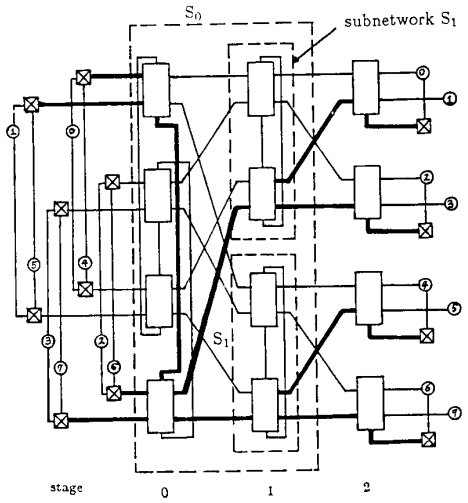

Fig. 2. The overall configuration of a completely chained network.

Definition 2: Any number of SE's in the same partition can form a *chain* by connecting the chain-in link of an SE to the chain-out link of itself or another SE within the same partition. A *complete chain* is a chain formed by connecting together all of the SE's within a partition. A network is called a *completely chained network* if all of its chains are complete chains.

The upper two SE's in stage 1 of Fig. 1 belong to a partition and so do the lower two SE's in the stage. Every chain in Fig. 1 is a complete chain because all SE's within a partition are connected together. According to Definition 2, SE's which constitute a chain can be connected in any order. A complete chain in stage 0 can also be connected as shown in Fig. 2. This connection avoids introducing unnecessary extra delay when an entire SE in stage 1 fails. A systematic approach to the construction of chains that can avoid extra delay due to an entire SE failure is given in [15].

Based on the recursive nature in the topology of a baseline network, the following theorem can be easily proved.

Theorem 1: At stage i, $0 \le i \le \log_2 N - 1$, there are 2^i partitions each with $N/2^{i+1}$ SE's.

From Theorem 1, we can obviously find out that 1) the number of complete chains in stage i is twice as many as those in stage $i-1$, for $0 < i < \log_2 N$; and 2) each SE in the last stage is a partition by itself.

B. Routing Procedure

Assume a network input, labeled by $i_0 i_1 \cdots i_t$, is to be connected to a network output, labeled by $d_0 d_1 \cdots d_t$. The routing algorithm in the "chained" baseline network is essentially the same as that used in a regular baseline network (i.e., the one introduced by Lawrie in [2]).

At stage i, bit d_i of the destination label is used to route a request in an "augmented" SE. If $d_i = 0$, it is routed to the upper output link of the SE; if $d_i = 1$, it is switched to the lower output link. If the destined output link is "blocked" due to a conflict, a link failure, or a failure in the SE at the next stage to which the output link is connected, the request is routed through the chain-out link to another SE within the chain. The same d_i bit will be utilized by an identical routing algorithm in the new SE. If the destined output link in the new SE is again "blocked," this request will be routed to yet another new SE in the same chain. A request can be routed through as many SE's within a chain as needed. Eventually a "good" output link can be found and the request proceeds to the next stage. Because an identical routing algorithm is employed in each SE, the request will try the same output link of SE's within a chain when it traverses the chain. Thus, we have the following theorem.

Theorem 2: The routing algorithm proposed above can route a request from an input $i_0 i_1 \cdots i_t$ to its destination $d_0 d_1 \cdots d_t$ through as many SE's within a chain as needed.

C. Overall Network Consideration

A chained network has alternative paths at every stage, so it is a strong reroutable network. When a request encounters a failure or a conflict at an SE, it can traverse an alternative path through the chain-out link. Neither a status table nor backtracking requests is needed for the rerouting process. Better yet, it uses the same routing tag and the same routing procedure.

The Achilles heel of this scheme is its input links of the first stage (i.e., stage 0) and the output links of the last stage (i.e., stage $\log_2 N - 1$). If any of those links fails, no matter what fault-tolerance scheme is employed inside a network, we still cannot gain access to the network.

To remedy this problem, several approaches similar to the one shown in Fig. 2 can be employed. Each external component now is allowed to have access to two input links of the first stage through extra switches. It is known from Theorem 1 that each SE in the last stage is itself a partition. By taking the chain-out link of an SE back to its chain-in link, we can improve network performance because a chain-in/chain-out pair acts as a buffer to hold a blocked request [12]. However, forming a chain with one SE does not provide any fault-tolerant capability. If we use extra switches and connect an external component to an output as depicted in Fig. 2, a single fault in the last stage can be tolerated (the failure of an extra switch is regarded as the failure of a chain-out link).

In a high-performance network design, the width of a link is not uncommon to be 32 or more. Due to power consumption and speed consideration, several sets of pins are usually needed to provide clock, power, and control in an SE. Hence, it is highly unlikely that a failure in an SE will disable the entire SE, especially when efforts are made in the design to avoid such an undesirable situation. A failure in an SE is here assumed to affect only its *partial* functionality. Each link coming in or going out of an SE has a *module* associated with it [8]. A module contains all of the control mechanism needed to route a request through the connecting link. A module is called an *input* (or *output*) *module* if its connecting link is an input (or output) link of an SE. A *chain-in* (or *chain-out*) *module* can be similarly named.

An *element* is formed by an output module of an SE, an input module of an SE in the subsequent stage, and the link connecting them. The chain-out module of an SE, its connecting link, and the chain-in module of the connected SE also constitute an element. An element in the first stage contains an input module and its connecting link. Likewise, an output module (or a chain-out module and its associated switch) together with the connecting link in the last stage also form an element. An *input (output) element* in stage i, $0 \le i < \log_2 N - 1$, is an element which contains an input (output) module of an SE in stage i. A *chain-in (chain-out) element* can be similarly defined. An element is considered to be *faulty* if any one of its components fails.

The fault model adopted in our subsequent analysis is that a fault disables an element instead of an SE or a link. An element is regarded as the *basic network unit* and a faulty element prevents a path involving that element from being realized. Also we consider only completely chained networks in our analysis since they have the maximum fault-tolerant capability.

III. Reliability Analysis

The following assumptions are made to facilitate our reliability analysis. 1) The event that an element becomes faulty is an independent event, and it occurs randomly. 2) A network is considered failed when the number and the locations of faulty elements prevent the connection of any path between an arbitrary input–output pair of the network.

An output element in stage i can be regarded as an input element in stage $i+1$, so we consider only output elements and chain-out elements in each stage along with the input elements of stage 0 when the total number of elements is counted. Since a completely chained

network has more than one path between each input–output pair, it can tolerate at least one faulty element. To calculate the upper bound on the number of tolerable faults in such a network is quite involved. First of all, let us consider the minimum number of fault-free elements required to provide full connectivity in the network.

A MIN comprises superimposed binary trees rooted at SE's in the first stage with SE's as their nodes and SE's in the last stage as their leaves. The binary tree with an SE in the first stage as its root is indicated by bold line segments in Fig. 2. A path from the root to every leaf exists if all elements along the tree are fault-free. Since an input link of the first stage is shared by two external components, it is sufficient for every external component to gain access to the network if one half of the input elements in stage 0 are fault-free. A path from every external component to the second stage is guaranteed, if the chaining elements needed to connect one half of SE's in the first stage as well as the two output elements of the rooted SE are fault-free (see Fig. 2). Therefore, $N/4 - 1$ chaining elements and two output elements in the first stage of a size N network must be functional. For middle stages, only two output elements in each chain are required and, from Theorem 1, there are 2^i chains at stage i, where $0 < i < n - 1$ ($n = \log_2 N$). An SE in the last stage can reach any one of the two attached external components if its chain-out elements are fault-free. From the above observation, we realize that the minimum number of fault-free elements required in a size N network is

$$N/2 + (N/4 - 1 + 2) + \left(\sum_{m=1}^{n-2} 2 \times 2^m \right) + N/2 = 9N/4 - 3. \quad (1)$$

The total number of elements in a network is $E = N + (3N \log_2 N)/2$. Hence, the maximum number of tolerable elements, denoted as B^u, is $N(3 \log_2 N - 5/2)/2 + 3$. A network with size 8 has $B^u = 29$, as depicted in Fig. 2.

The number given by B^u is good for a completely chained network in which to backtrack requests is allowed or status tables are maintained, i.e., without strong reroutability. To a completely chained network with strong reroutability, let us obtain a conservative upper bound, denoted as B, based on the following assumption. Two faulty elements will cause a chained network to fail if 1) both faults are at the same SE and 2) one of them is an output element and the other is the chain-out element. The assumption is quite conservative because some SE's in a stage other than the last one can fail totally (i.e., the two output elements and the chain-out element simultaneously fail) without ruining the full connectivity of the network. For example, if an SE is not reachable due to failures in previous stages and in its chain-in element, then the SE is allowed to be totally faulty. The exact number of tolerable faults in a chain depends on the fault pattern existing in previous stages. Nonetheless, this assumption enables us to calculate the survival probability of chains, from which a conservative network survival probability can be derived. We have a substantial reliability improvement on a completely chained network even under this conservative assumption, as will be shown.

A complete chain with $M/2$ SE's has M output elements and $M/2$ chain-out elements. The worst case a chained network can tolerate is when all but two of the output elements in the complete chain are faulty, and all but one of the chain-out elements are good. Thus, a complete chain can tolerate at most $(M - 2) + 1 = M - 1$ faulty elements. From Theorem 1, stage i, $0 \le i \le \log_2 N - 1$, has 2^i complete chains each with $N/2^{i+1}$ SE's. For stages between 0 and $\log_2 N - 2$, the largest possible number of faults a network can tolerate is

$$\sum_{i=0}^{\log_2 N - 2} 2^i \times (N/2^i - 1) = N(\log_2 N - 3/2) + 1.$$

Each external component has access to two input elements in stage 0, so at most one half of the input elements in the stage can be tolerated. In the last stage, each SE can tolerate two faulty output elements, so up to N total faults are tolerable. The number of faulty elements the entire network can tolerate is $B = N(\log_2 N - 3/2) + 1 + N/2 + N = N \log_2 N + 1$.

A. Expected Number of Tolerable Faults

Faults in a network actually take place at random. In the following analysis, the number of faults which cause a network to fail is assumed to be a random variable. The previous conservative assumption is also used here. To simplify our analysis, we divide the elements of a network into three portions: 1) the input elements of the first stage, 2) the output elements and the chain-out elements in stages 0 through $\log_2 N - 2$, called intermediate stages, and 3) the output and the chain-out elements in the last stage.

First, let us consider the nth complete chain in an intermediate stage s, denoted by C_{sn}, where $0 \le s \le \log_2 N - 2$ and $1 \le n \le 2^s$. Assume $q_{sn}(i)$ is the probability that C_{sn} can still provide connections to the next stage after having i faulty elements. Then $q_{sn}(i)$ is expressed as a function of N, s, and i [10]. With $q_{sn}(i)$'s, we can derive the *survival probability* of intermediate stages after having k faulty elements by an iterative method. As shown in Fig. 2, a *subnetwork* S_j, $0 \le j \le \log_2 N - 3$, consists of three parts: a complete chain in stage j and two S_{j+1}'s. Note that an $S_{\log_2 N - 2}$ contains a complete chain with two SE's in stage $\log_2 N - 2$. Assume i_{j1}, i_{j2}, and i_{j3} are the number of faulty elements in each of those three parts in S_j. Let I_j^k be a triplet $\langle i_{j1}, i_{j2}, i_{j3} \rangle$ such that $i_{j1} + i_{j2} + i_{j3} = k$, and $D_j(I_j^k)$ be the probability that a fault pattern I_j^k will occur in subnetwork S_j. Since each fault pattern is of equal probability, we have

$$D_j(I_j^k) = \frac{\binom{L_j^1}{i_{j1}} \binom{L_j^2}{i_{j2}} \binom{L_j^3}{i_{j3}}}{\binom{L_j}{k}}$$

where L_j is the total number of elements in subnetwork S_j, and L_j^1, L_j^2, L_j^3 are the total number of elements in each of those three parts in S_j, respectively, with $L_j^1 + L_j^2 + L_j^3 = L_j$. It is obvious that $L_j^1 = L_j/(\log_2 N - 1 - j)$ and $L_j^2 = L_j^3 = L_{j+1}$ with the boundary condition $L_{\log_2 N - 2} = 6$. Let $Q_j(k)$ be the probability that subnetwork S_j can still function after having k faulty elements, then

$$Q_j(k) = \sum_{I_j^k \in U_j^k} D_j(I_j^k) q_{jn}(i_{j1}) Q_{j+1}(i_{j2}) Q_{j+1}(i_{j3}) \quad (2)$$

where $U_j^k = \{ I_j^k \mid I_j^k = \langle i_{j1}, i_{j2}, i_{j3} \rangle, i_{j1} + i_{j2} + i_{j3} = k \}$, and $Q_{\log_2 N - 2}(i) = q_{sn}(i)$ with $s = \log_2 N - 2$. We can compute (2) starting from stage $\log_2 N - 3$, and work all the way back to stage 0. After $\log_2 N - 2$ steps, we can obtain $Q_0(k)$, the survival probability of intermediate stages with k faulty elements.

Next, let us consider the survival probability for the input elements of the first stage and the survival probability of the last stage. Let $Q_f(k)$ be the probability that k input elements in stage 0 are faulty but they do not cause the network to fail; and $Q_l(k)$ be the survival probability of the last stage, i.e., stage $\log_2 N - 1$, after having k faulty elements. The expressions for $Q_f(k)$ and $Q_l(k)$ can be found in [10].

Having obtained $Q_f(k)$, $Q_0(k)$, and $Q_l(k)$, we can compute $Q(k)$, the survival probability of the entire network after k elements fail using similar steps in deriving (2).

With $Q(k)$'s, we can compute the probability $P(k)$ that k or fewer faults cause the network to lose full connectivity:

$$P(k) = 1 - Q(k). \quad (3)$$

Let $p(i)$ be the probability that the ith fault will cause the network to fail. We get

$$P(k) = \sum_{i=2}^{k} p(i). \quad (4)$$

TABLE I
$\bar{k}$, $MTTF_0$, MTTF, AND COST-EFFECTIVENESS RATIO OF NETWORKS

N	$\bar{k}$	$MTTF_0$	$MTTF$	$MTTF/MTTF_0$	μ/μ_0
4	4.5	$(8.0\lambda)^{-1}$	$(3.0\lambda)^{-1}$	2.7	1.2
16	11.5	$(53.3\lambda)^{-1}$	$(9.2\lambda)^{-1}$	5.8	2.6
64	27.2	$(298.7\lambda)^{-1}$	$(22.9\lambda)^{-1}$	13.0	5.8
256	61.8	$(1536.0\lambda)^{-1}$	$(53.2\lambda)^{-1}$	28.9	12.8
1024	137.1	$(7509.3\lambda)^{-1}$	$(118.9\lambda)^{-1}$	63.2	28.1

From (3) and (4), we can obtain all of $p(i)$'s recursively. Now, let $\bar{k}$ be the expected number of faulty elements that cause the network to lose full connectivity under the conservative assumption given above, then

$$\bar{k} = \sum_{i=2}^{B+1} ip(i)$$

where $B = N \log_2 N + 1$ is the conservative upper bound we obtained previously. Notice that $p(i) = 0$ for $i > B + 1$ because the network fails as $i = B + 1$. The k for a variety of network sizes N is given in Table I. It is interesting to see that as N increases, $\bar{k}$ also grows. The larger a network is, the better a completely chain network can survive from more faulty elements (because more redundant paths are provided). This is a sharp contrast to a regular network with only one unique path between each input-output pair. A strong reroutable fault-tolerant network [5], [10], [11] always possesses this property.

B. Mean Lifetime of a Network

The mean time to fail (MTTF) for completely chained networks is derived based on the assumptions that failures occur independently to elements with a constant rate λ. Let $R(t)$ be the reliability function of a chained network, then

$$R(t) = \sum_{i=0}^{E} Q(i) \binom{E}{i} (e^{-\lambda t})^{(E-i)} (1 - e^{-\lambda t})^i$$

where $E = N + (3N \log_2 N)/2$ is the total number of elements in the network. Under our conservative assumption, $Q(i)$ is 0 as $i > B$. The MTTF of the network can be obtained after a simple calculation as $\text{MTTF} = (1/\lambda) \sum_{i=0}^{E} (Q(i))/(E-i)$.

To understand the reliability improvement of a completely chained network over a regular network, we compare the two networks with the same size. Since a 2×2 "augmented" SE in a chained network is essentially a 3×3 crossbar switch, the hardware complexity of an element in a 2×2 "augmented" SE is roughly 3/2 times as complicated as that of an element in a regular 2×2 SE. Let us assume the failure rate to an element of a regular 2×2 SE be $(2/3)\lambda$ due to lower hardware complexity (comparing to λ for an element in a 2×2 "augmented" SE). A regular network fails after a single fault arises because there is only one unique path between each input-output pair. Its mean time to fail, denoted by MTTF_0, for various network sizes is given in Table I, where the corresponding MTTF is also listed. From the table, we can see that a large completely chained network performs far more reliably than a regular network. This is because larger completely chained networks have larger chains (i.e., chains with more SE's) and can thus provide more alternative paths. Their reliability is significantly increased even under the conservative assumption.

Now, a simple measure of cost effectiveness is given for comparison between chained networks and regular networks. An "augmented" SE has higher hardware complexity and, hence, higher cost than a regular SE. We assume the hardware cost of an SE is proportional to the number of *crosspoints* within an SE. The hardware cost of a 2×2 regular SE and an "augmented" SE is four units and nine units, respectively. Since a chained network and a regular network of the same size employ equal number of SE's, the hardware cost of a chained network is 9/4 times that of a regular network (ignoring the cost of extra switches in a chained network to connect external components). Let us define the cost-effectiveness measure μ of a network as the ratio of its MTTF to cost. In the last column of Table I, we list the ratio of μ, the measure for a chained network, to μ_0, the measure for a regular network. It can be seen that the ratio is always greater than 1 and grows rapidly as the network size increases. A chained network is more cost effective than a regular one.

IV. Discussion and Comparison to Other Schemes

A chained network can provide fault tolerance as long as every chain in the network contains more than one SE. A fault-tolerant chained network is strong reroutable if every SE belongs to a chain (except SE's in the last stage). Even though we study only completely chained networks here because they provide the maximum fault-tolerant capability, other chaining configurations are also possible. For example, if pairs of SE's within a partition are chained together, it becomes Augmented Bidelta Network as proposed in [11].

A regular MIN exhibits unsatisfactory bandwidth performance because an internal link is shared by many paths. Request conflicts become increasingly serious as the network size grows. A scheme to pair two SE's in a stage to share load and to balance traffic is recently introduced in [13], where some implementation issues and network performance are detailed. Based on the same reason, a chained network enjoys the byproduct of bandwidth improvement as presented in [15].

An extra stage cube (ESC) network [6] is proposed to provide single-fault tolerance. Compared to a regular network built from $k \times k$ SE's, the hardware overhead of an ESC network with $k \times k$ SE's is $1/\log_k N + O(1/\log_k N)$; and it is $(2/k + 1/k^2) + O(1/\log_k N)$ for a baseline network with the proposed chaining scheme using $(k + 1) \times (k + 1)$ SE's, where N is the network size. The first term of these two expressions is attributed, respectively, to the extra stage in an ESC network and to the $(k + 1) \times (k + 1)$ SE's (i.e., "augmented" SE's) in a chained network. The second term is attributed to the extra multiplexers/demultiplexers. From the expressions, we realize that an ESC network has less hardware overhead only when k is small or N is very large. If we ignore the second term in the above expressions (i.e., the overhead due to multiplexers/demultiplexers), for $k = 8$, an ESC network has less hardware overhead only when N is larger than 2048; for $k = 16$, it has less overhead only when N is larger than 2^{31}.

In many previous fault-tolerance schemes, alternative paths are provided through the links *between* consecutive stages by adding extra stages [6], [9] or by using larger SE's in certain stages [8]. They provide a *fixed* number of alternative paths for any sized network. By contrast, the alternative paths in a chained network are provided through the links within the *same* stage. A chained network exploits all inherent paths in the tree structure embedded in the network. Hence, the number of alternative paths between each input-output pair grows exponentially as the network size increases. Although its alternative paths are not totally disjoint, a chained network is more reliable than a network with only a fixed number of alternative paths as the system size grows. This is illustrated below by the terminal reliability of various networks. The terminal reliability between a given network input-output pair is defined as the probability of existence of at least one path between them.

Recall that an element is considered to be the basic unit in our fault model. The reliability of an element is assumed to be a function of time $e^{-\lambda t}$ in deriving network MTTF previously. Now, suppose an element has constant reliability r instead, which takes the failure rate of both switching logics and the connecting link into account. To simplify our analysis, we assume the reliability of an input element at the first stage and an output element at the last stage is r as well. The network under consideration is of size $N = 2^n$. In an ESC network [6] or the Indra network [9] built from 2×2 SE's (i.e., $R = 2$), a path from S (a source node) to D (a destination node) consists of $n + 2$ elements, including the two elements for connecting S and D. The terminal reliability between S and D in an ESC network is $r^2(1 - (1$

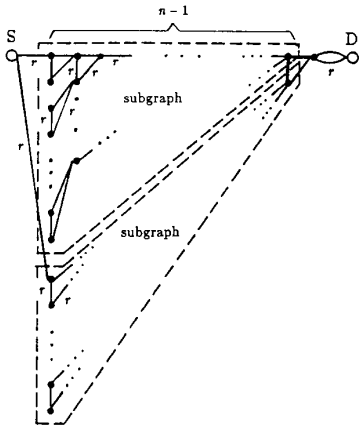

Fig. 3. The redundancy graph of a network constructed by chaining together two SE's in each partition.

TABLE II
TERMINAL RELIABILITY OF VARIOUS NETWORKS WITH $r = 0.9$

Network size (N)	terminal reliability			
	ESC	Gamma (max.)	Indra ($R=2$)	chained network (TR_m)
4	0.781	0.875	0.963	0.969
16	0.714	0.844	0.925	0.961
64	0.632	0.815	0.869	0.951
256	0.547	0.787	0.798	0.939
1024	0.466	0.759	0.717	0.926
4096	0.393	0.733	0.633	0.912

$- r^n)^2$). The terminal reliability of the INDRA network is derived in [9], where an SE is treated as an entity with a given reliability value (which is different from our fault model) and a path from S to D is composed of $n + 1$ nodes. Thus, in citing the results, we have to be cautious about the slight difference caused by different fault models.

To calculate the terminal reliability of a completely chained network, denoted as TR, is very involved. Instead let us consider the terminal reliability of a network in which pairs of SE's within a partition are chained together. The redundancy graph [8] for a source-destination pair in such a chained network is depicted in Fig. 3, where a reliability value is associated with each link. Suppose its terminal reliability is represented by TR_m. It is obvious that TR_m = TR for $N = 4$ and $TR_m <$ TR for $N > 4$. The reliability of subgraphs indicated in Fig. 3 is $g = r(1 - (1 - r)(1 - r^2))^{n-2}$. Now the terminal reliability between S and D can be derived by applying the bridge network result presented in [14]. We have

$$TR_m = 2gr - (gr)^2 + 2(1-g)(1-r)gr^2(2r-r^2).$$

The terminal reliability of various fault-tolerant networks with $r = 0.9$ is summarized in Table II. The second column gives the terminal reliability of an ESC network. The maximum terminal reliability of a Gamma network [5] is taken from [9] with appropriate adjustment to reflect the difference of fault models; so is the terminal reliability of an Indra network with $R = 2$. The actual terminal reliability of a completely chained network (i.e., TR) is greater than or equal to figures shown in the last column. It can be found that a chained network always possesses a higher terminal reliability. The larger the network size is, the more a chained network outperforms other networks. Similar arguments can be applied to a modified omega network [8] as well.

V. CONCLUDING REMARKS

We propose a simple fault-tolerance enhancement scheme which can be applied to a wide class of multistage interconnection networks. Network fault-tolerant capability is greatly enhanced by chaining switching elements within the same stage together to provide multiple paths between each network input-output pair. It takes full advantage of the tree structure embedded in a network, so the number of alternative paths increases exponentially as the system size grows. A completely chained network possesses strong reroutability which permits a request to be rerouted by a simple routing procedure within the same stage when the request encounters a failure or a conflict. From the reliability analysis, we show that the network reliability is enhanced greatly as the network size becomes larger. It is also illustrated that a pair of source-destination nodes in a chained network has higher terminal reliability than in many other fault-tolerant networks.

The network bandwidth of a chained network is fundamentally improved due to the existence of buffering capability, as detailed in [15]. The behavior of a chained network with a single fault is also presented there. Although a fault can affect many paths, it is shown to have very little impact on bandwidth of a chained network, i.e., it performs almost equally well before and after the occurrence of the fault. For a network with buffers in its SE's, the chaining scheme is demonstrated [15] to be able to improve network delay effectively through dynamically regulating traffic among alternative paths to eliminate local congestion.

REFERENCES

[1] C.-L. Wu and T.-Y. Feng, *Tutorial: Interconnection Networks for Parallel and Distributed Processing*. IEEE Computer Science Press, 1984.

[2] D. H. Lawrie, "Access and alignment of data in an array processor," *IEEE Trans. Comput.*, vol. C-24, pp. 1145-1155, Dec. 1975.

[3] C.-L. Wu and T.-Y. Feng, "On a class of multistage interconnection networks," *IEEE Trans. Comput.*, vol. C-29, pp. 694-702, Aug. 1980.

[4] J. P. Shen and J. P. Hayes, "Fault tolerance of a class of connecting networks," in *Proc. 7th Annu. Symp. Comput. Architecture*, 1980, pp. 61-71.

[5] D. S. Parker and C. S. Raghavendra, "The gamma network: A multiprocessor interconnection network with redundant paths," in *Proc. 9th Annu. Symp. Comput. Architecture*, Apr. 1982, pp. 73-80.

[6] G. B. Adams III and H. J. Siegel, "The extra stage cube: A fault-tolerant interconnection network for supersystems," *IEEE Trans. Comput.*, vol. C-31, pp. 443-454, May 1982.

[7] J. E. Lilienkamp, D. H. Lawrie, and P.-C. Yew, "A fault tolerant interconnection network using error correcting codes," Dep. Comput. Sci. Rep. UIUCDCS-R-82-1094, Univ. Illinois Urbana-Champaign, June 1982.

[8] K. Padmanabhan, "Fault tolerance and performance improvement in multiprocessor interconnection networks," Ph.D. dissertation, Dep. Comput. Sci. Rep. UIUCDCS-R-84-1156, Univ. Illinois, Urbana-Champaign, May 1984.

[9] C. S. Raghavendra and A. Varma, "INDRA: A class of interconnection networks with redundant paths," in *Proc. Real-Time Syst. Symp.*, Dec. 1984, pp. 153-164.

[10] N.-F. Tzeng, P.-C. Yew, and C.-Q. Zhu, "A fault-tolerant scheme for multistage interconnection networks," in *Proc. 12th Int. Symp. Comput. Architecture*, June 1985, pp. 368-375.

[11] V. P. Kumar and S. M. Reddy, "Design and analysis of fault-tolerant multistage interconnection networks with low link complexity," in *Proc. 12th Int. Symp. Comput. Architecture*, June 1985, pp. 376-386.

[12] D. M. Dias and J. R. Jump, "Analysis and simulation of buffered delta networks," *IEEE Trans. Comput.*, vol. C-30, pp. 273-282, Apr. 1981.

[13] C.-T. A. Lea, "The load-sharing banyan network," *IEEE Trans. Comput.*, vol. C-35, pp. 1025-1034, Dec. 1986.

[14] P. M. Lin, B. J. Leon, and T.-C. Huang, "A new algorithm for symbolic system reliability analysis," *IEEE Trans. Reliability*, vol. R-25, pp. 2-15, Apr. 1976.

[15] N.-F. Tzeng, "Fault-tolerant multiprocessor interconnection networks and their fault-diagnoses," Ph.D. dissertation, Dep. Comput. Sci., Univ. Illinois, Urbana-Champaign, Aug. 1986.