
The Journal of Supercomputing, 34, 63–80, 2005

Designing A Disjoint Paths Interconnection Network with Fault Tolerance and Collision Solving*

CHING-WEN CHEN‡ chingwen@mail.cyut.edu.tw
Department of Computer Science and Information Engineering, Chaoyang University of Technology, Wufeng, Taichung County, Taiwan 413, ROC

CHUNG-PING CHUNG cpchung@csie.nctu.edu.tw
Department of Computer Science and Information Engineering, National Chiao Tung University, Hsinchu, Taiwan 300, ROC

Abstract. In fault-tolerant interconnection designs, many prior researches suggest good use of disjoint paths to improve the reliability of interconnection networks. Although disjoint paths increase reliability, they always cost the throughput penalty. To address the problems of both performance and fault-tolerant capability, the following issues should be carefully considered: (1) guarantee of at least two disjoint paths, (2) easy rerouting between disjoint paths, (3) keep low rerouting hops, (4) solve the occurrences of packets' collision. In this paper, we consider these issues to design a fault-tolerant network called CSMIN (Combining Switches Multistage Interconnection Network). CSMIN provides two disjoint paths to guarantee one fault-tolerant and can dynamically reroute packets between these two paths to solve the collision situation. In other words, to switch packets between these two disjoint paths easily, CSMIN causes these two disjoint paths to have regular distances at each stage. Accordingly, a packet can be dynamically sent to the other disjoint path if it encounters a faulty or busy element. In addition, CSMIN presents low rerouting hops (an average of one rerouting hop) to maintain a low collision ratio. From the simulation result, CSMIN performs with a better arrival ratio than Gamma and other related disjoint paths networks do.

Keywords: multistage interconnection networks, gamma interconnection networks, disjoint paths, dynamic rerouting, rerouting hops, collision ratio

1. Introduction

Multistage interconnection networks (MINs) are well suited for communications among tightly coupled system components, and offer a good balance between cost and performance. For complex systems, assuring high reliability is a significant task [1]. In addition, the performance of interconnection networks has a great impact on system latency and throughput of parallel systems. Thus fault-tolerance and high performance are crucial for MINs serving the communication needs of large-scale multiprocessor systems.

To enhance fault-tolerant capability, some researchers suggested that 3×3 switches be used as basic building blocks [2–10, 12, 13]. The Gamma interconnection network (GIN)

*This research was supported by the National Science Council NSC-91-2218-E-324-006.
‡To whom all correspondence should be addressed.

[9], augmented data manipulator (ADM), and inverse augmented data manipulator (IADM) [8] are in this category. The GIN provides multiple paths between any source-target pair except when the indices of the source and target are the same. To improve the fault-tolerant capability of GIN, some researchers modified GIN to provide multiple paths between any source-target pair [6, 10, 13]. For example, the extra stage gamma network [13] and the PM22I interconnection network [6] add an extra stage to the GIN to provide multiple paths between any source-destination pair. However, the method of adding extra stage results in more hardware cost and more collisions. Although monogamma interconnection network (MGIN) [10] provides multiple paths between any source-target pair with almost hardware cost than GIN, it cannot guarantee there exists at two disjoint paths between any source-destination pair. Some researchers derived Gamma networks to have at least two disjoint paths between any source-destination pair [2–4, 12] with almost hardware cost than the GIN. Examples of these networks include CGIN [3], composite banyan [12], PCGIN [2], and Balanced Gamma Interconnection Network (BGIN) [4]. In contrast with providing disjoint paths, B-network [5, 7], which modified the GIN, provides the capability of dynamic rerouting to prevent the collisions during the routing path. However, B-network cannot guarantee one fault tolerance.

The networks that were mentioned above provide at least two disjoint paths between any source-destination pair to increase reliability, but they do not consider how to use disjoint paths to handle collision problems in order to obtain good performance (throughputs). Hence, the penalties in extra hardware cost or number of rerouting hops for switching a packet from one disjoint path to another remain unknown. Besides, the methods of using extra stages increase the collision ratio.

Consequently, when a disjoint paths network is designed to tolerate faults and deal with packet collisions, the following essentials should be carefully considered.

(1) Guarantee that at least two disjoint paths are provided at each stage.
(2) Have the capability of easy rerouting from one disjoint path to another path at each stage.
(3) Keep low rerouting hops.
(4) Solve the occurrences of packet collisions.

In this paper, we present a fault-tolerant network called CSMIN with two disjoint paths between any source-destination pair to guarantee one fault-tolerance. Besides, to reroute packets between these two disjoint paths to deal with encountering a faulty or busy element, these two paths are designed to have 2^i switches at stage i such that the switches can easily reroute the packets from one path to the other. Moreover, the average number of rerouting hops is one. Because of lower rerouting hops, our network performs with a better arrival ratio than other related networks. The simulation result shows these results.

The rest of this paper is organized as follows. In Section 2, we introduce the topology and problems of GIN. In Section 3, we present our design, which is called Combining Switch Multistage Interconnection Network (CSMIN), with two disjoint paths and a way to reroute a packet from a current path to the other disjoint path. Section 4 presents the results of our

experimental result. Finally, Section 5 describes this paper's contributions and offers ours conclusions.

2. Gamma interconnection networks and related disjoint paths networks

In this section, we introduce the topology, routing methods and properties of Gamma networks. In addition, we also point out the fault-tolerant problems in Gamma networks and related multistage interconnection networks with disjoint paths.

2.1. Gamma interconnection network (GIN)

2.1.1. Topology. A Gamma network of size $N = 2^n$ consists of $n + 1$ stages labeled from 0 to n. Each stage has N switches labeled from 0 to 2^n–1 [9]. The switch architecture at the first and the last stages has 1×3 and 3×1 crossbars, respectively, and switches located at the intermediate stages have 3×3 crossbars. A switch labeled j at stage i has three output links connecting to switches at stage $(i + 1)$ based on the plus-minus-2^i function; that is, the jth switch at stage i has three output links connected to switches $[(j - 2^i) \bmod N]$, j, and $[(j + 2^i) \bmod N]$ at the next stage. Figure 1 illustrates a GIN network of size 8.

2.1.2. Multiple paths. In GIN, an n-bit tag determines the routing path connecting the source S to the target T. Each tag bit can be 1, 0, or $\bar{1}$. An n-bit tag D expresses the difference between target T and source S, i.e., $D = T - S \pmod N$. Bit d_i is used at stage i in such a way that the lower/upper connection is taken if d_i is equal to $1/\bar{1}$, and the straight connection is taken if d_i is 0 where the distance $D = d_0 d_1 \ldots d_{n-1}$. In Gamma networks,

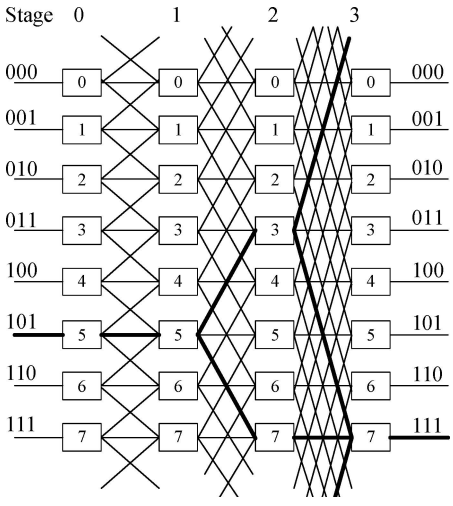

Figure 1. Gamma Interconnection Network with $N = 8$ and three paths between nodes 5 and 7.

a non-zero tag D has multiple representations, indicating there are multiple paths between source S and target T, when $S \neq T$. For example, when $N = 8$, $S = 5$ and $T = 7$, the tag D can be 010 or $0\bar{1}1$, or $0\bar{1}\bar{1}$. The three corresponding paths are shown again in Figure 1.

2.2. Problems with gamma networks and previous works on providing disjoint paths

Although Gamma networks provide distance tag routing and multiple paths, they cannot guarantee one fault-tolerance. Gamma networks have only one single path when the indices of the source and the target are the same. Furthermore, Gamma networks present multiple paths, but they cannot guarantee that these paths are disjoint because when the distance between the source and the target is even, the straight link between stage 0 and stage 1 and the switch at stage 1 connected by the straight link is the common element contained in these paths. For example, in Figure 1, there are three paths when the source is 5 and the target is 7, but these paths are not disjoint because they use the common switch 5 at stage 1. Accordingly, once switch 5 at stage 1 is faulty, there is no path to deliver packet from source 5 to target 7.

There are many previous works providing disjoint paths to guarantee fault-tolerance [2–4, 12]. For example, these networks include CGIN [3], composite banyan [12], PCGIN [2], and Balanced Gamma Interconnection Network (BGIN) [4]. The BGIN [2] and the composite banyan [12] modified the redundant link (one of two non-straight links of the switches in the GIN) to a symmetric link to generate two disjoint paths between any source-target pair. With regard to CGIN [3], the network copies the links between the first two stages to the links between the last two stages to generate two disjoint paths that are parallel during the middle stages. Finally, PCGIN [2] adds one link to the switches at stage 0 to generate two disjoint paths between any source-destination pair. However, these networks use two methods to handle the situation of a packet encountering a faulty or busy element. One method sends two identical packets concurrently from the source to the destination along the two disjoint paths. This method causes more packet collisions. The other method uses backtracking rerouting [12]. The backtracking method is that a switch to send a packet back along the traversed path to the source, and takes another disjoint path to tolerate the faulty element. However, if the backtracking scheme is applied, all output links in a switch are changed to bi-direction and the rerouting hop count is high. As a result, the hardware cost and collision rate increase. About methods of using extra stages to tolerate faults, the methods suffer the increasing the hardware cost and the collision rate because the length of routing paths increases no matter whether packets encounter a faulty or busy element or not.

In our paper, to guarantee one fault-tolerance and have easily rerouting capability between disjoint paths with low rerouting hops, we present, in Section 3, a network called Combining Switches Multistage Interconnection Network (CSMIN) and show its fault-tolerance capability and how to reroute packets between these two disjoint paths.

3. Combining switches multistage interconnection network (CSMIN)

In this section, we propose a network called Combining Switches Multistage Interconnection Network (CSMIN) that guarantees one fault-tolerance by providing two disjoint paths for

any source and target pair. Besides, the CSMIN can also reroute a packet between these two disjoint paths at each stage when a packets encounters a faulty or busy element, with an average of one rerouting hop. Section 3.1 describes the topology of the CSMIN. Section 3.2 presents the two disjoint paths in the CSMIN. Section 3.3 describes how the CSMIN route and reroute packets between these two disjoint paths.

3.1. Topology of CSMIN

The topology of CSMIN is described as follows:

1. At stage 0, switch $2i$ and switch $2i + 1$ are coupled into a 2×4 switch, for $i = 0$ to $(N/2 - 1)$.
2. All straight links between stage 1 and stage n are bi-directional.
3. All the rest remain the same as those in GIN.

In CSMIN, the sizes of switches of the first and last stages are 2×4 and 3×2, respectively. Switches at stage 1 have 3×3 crossbars. Moreover, each switch located at the intermediate stages has a 4×4 crossbar, as shown in Figure 2.

3.2. Two disjoint paths in CSMIN

In this section, we present two disjoint paths in CSMIN which have a distance 2^i at stage i. We define two distance tag routing functions as *Downward* and *Upward*. The path generated

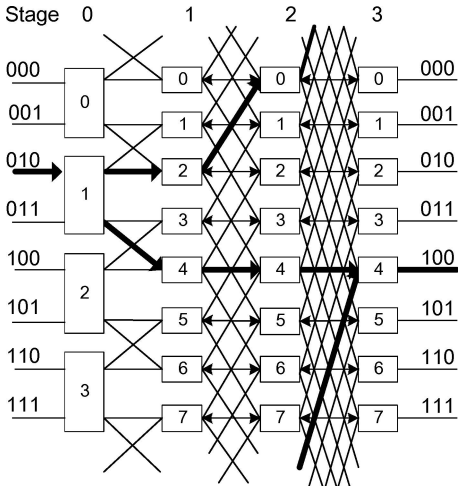

Figure 2. The two disjoint paths in CSMIN with source = 2 and target = 4.

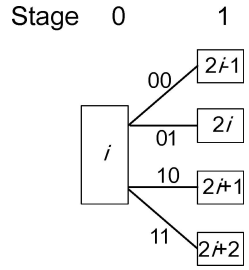

Figure 3. The four routing links and two routing bits from stage 0 to stage 1.

by the *Downward* function always goes through the downward non-straight or straight links only. The *Upward* function is the route consisting of only the upward non-straight or straight links. At stage 0, the four conditions (00, 01, 10, 11) corresponding to the four links are as shown in Figure 3. We describe the algorithm to generate the *Downward* and *Upward* tags in Algorithm 1.

Algorithm 1: Generating the routing tags of two disjoint paths in CSMIN

1. Input source S and target T.
2. If $T - S$ is even
   If S is even
   $S = S + 1$
   Else
   $S = S - 1$
   Endif
3. Let the distance $DownwardD = (T - S)$ Mod N, and $UpwardD = (N - (T - S))$ Mod N.
4. Convert $DownwardD$ to its binary representation $c_0c_1c_2 \ldots c_{n-1}$.
5. Convert $UpwardD$ to its binary representation $b_0b_1b_2 \ldots b_{n-1}$ and replace any 1 with $\overline{1}$.
6. If $c_0 = 0$ and S is odd/even

   Replace c_0 with $c_{00}c_{01} = 10/11$ and b_0 with $b_{00}b_{01} = 00/01$

7. If $c_0 = 1$ and S is odd/even

   Replace c_0 with $c_{00}c_{01} = 11/10$ and b_0 with $b_{00}b_{01} = 01/00$

8. Output *Downward* tag $c_{00}c_{01}c_1c_2 \ldots c_{n-1}$ and *Upward* tag $b_{00}b_{01}b_1b_2 \ldots b_{n-1}$.

The *Downward* function is characterized by motion either in the straight or downward direction, and the *Upward* function goes upward or straight. An example illustrating two disjoint paths in the CSMIN with $S = 2$ and $T = 4$ is shown in Figure 2.

Theorem 1 *The distance between the disjoint paths generated by the Downward and Upward functions is 2^i at stage i, where $1 \leqq i \leqq n-1$.*

Proof: By Algorithm 1, the sum of *DownwardD* and *UpwardD* is $N (=2^n)$. Let the *Downward* tag be $c_{00}c_{01}c_1c_2 \ldots c_{n-1}$ and the *Upward* tag be $b_{00}b_{01}b_1\ b_2 \ldots b_{n-1}$.

Let i be the least number from 1 to $n-1$ such that b_i is different from c_i. Because of 2's complement property, 2^i is added to the vertical distance between the two disjoint paths from stage i to stage $i+1$ for $1 \leqq i \leqq n-2$. The distance between the *Downward* and *Upward* paths at stage 1 is 2 by Algorithm 1. Thus, the vertical distance at stage i between these two disjoint paths is $2 + 2^1 + 2^2 + \ldots 2^{i-1} = 2^i$ where $2 \leqq i \leqq n-1$.

Hence, the vertical distance between the *Downward* and *Upward* paths at stage i is 2^i, where $1 \leqq i \leqq n-1$. □

Theorem 1 explains the distance between the *Downward* and *Upward* paths at each stage. In Theorem 2, we prove that the two paths, *Downward* and *Upward*, are disjoint.

Theorem 2 *The two paths generated by the Downward and Upward functions are disjoint paths.*

Proof: By Theorem 1, the vertical distance between the *Downward* and *Upward* paths at stage i is 2^i. The two paths generated by the *Downward* and *Upward* functions are disjoint from stage 1 to stage $n-1$. At stage n, the two paths arrive at the target. Thus, the two paths are disjoint paths. □

From Theorem 1 and 2, we get two disjoint paths in CSMIN, which has a distance 2^i at stage i. Therefore, the network (CSMIN) can tolerate one fault by providing two disjoint paths. However, the backward straight links are not used until now. In the next section, we present how to use the backward straight link to reroute packets between these two disjoint paths once encountering a faulty or busy element.

3.3. Routing and rerouting in CSMIN

In Section 3.2, we have proved that there are two disjoint paths for any source and destination pair. In this section, we present how to use the two disjoint paths to reroute the packets. To dynamically switch packets between these two paths, the reversed straight link is used to reroute packets in the CSMIN. In Section 3.3.1, the situation of routing packets without a faulty or busy element is presented (without rerouting behaviors). In Section 3.3.2, we present the rerouting behaviors when the packets encounter a faulty or busy element in CSMIN. In addition, we also discuss and analyze the differences between these two situations.

3.3.1. Routing without a faulty or busy element. In this section, we describe the routing behavior without rerouting in CSGIN. If a packet does not encounter a faulty or busy element, distance tag routing is applied in CSMIN; that is, we compute the routing tags,

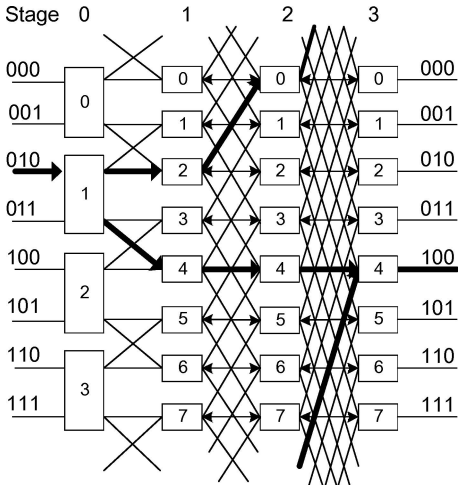

Figure 4. The two disjoint paths, (2, 4, 4, 4) and (2, 6, 0, 4), indicated by the bold line in CSMIN with source = 4 and target = 4 where the 4-tuple items mean the switch indices at stage 0, 1, 2, and 3 respectively.

the *Downward* and *Upward* tags, and then one of the two routing tags is used to send packets to the destination. Example 1 illustrates the routing path for the source $S = 4$ and the destination $T = 4$ with size $N = 8$.

Example 1 If source $S = 4$, destination $T = 4$, and network size $N = 8$, the routing situation in CSMIN is as shown in Figure 4.

Solution:

1. Because $S = 4$, $T = 4$, $T - S = 0$ and S is even, let $S = S + 1$ (=5) by Algorithm 1.
2. $DownwardD = (T - S) \bmod N = 7$ and $UpwardD = N - DownwardD = 1$.
3. Convert $DownwardD$ to obtain the *Downward* tag ($c_{00}c_{01}c_1c_2 = 1111$).
4. Convert $UpwardD$ to its binary representation to obtain the *Upward* tag ($=b_{00}b_{01}b_1b_2 = 0100$).
5. Take one of these two tags as the main routing tag.

3.3.2. Rerouting method with faulty and busy elements. In this section, we introduce the rerouting methods when a faulty element and a busy element are encountered. If an element is faulty, we assume that the pre-stage switch has the information to decide whether to route or reroute a packet for the conditions of partially or fully faulty switches. The rerouting behavior for encountering a faulty element occurs before the location of the faulty element. In contrast, if a packet collides with other packets at a switch, the switch reroute these packets dynamically. If a switch at stage i routes a packet to a faulty output link or to a faulty switch at stage $i + 1$, the switch at stage i reroute the packet to avoid encountering the faulty element. If there are two packets in a switch at stage i are delivered to the same

output link, the switch reroutes one of these two packets to solve the collision. Figures 5 and 6 show these two conditions mentioned above.

If the switch needs to reroute the packets, there are two methods to get the rerouting tags: 1. One is to bring two routing tags, *Downward* and *Upward*, before sending it. 2. The switch computes the new routing tag by a $2'$ complement computation. Therefore, the switch sends the rerouted packets from one disjoint path to the other disjoint path. Therefore, the new routing tag (or the other routing tag) is used for further routing. Algorithm 2 shows the method of computing rerouting tags in the switch.

Algorithm 2: Generating rerouting tag

The original *Downward* tag $= c_{00}c_{01}c_1c_2 \ldots c_{n-1}$

The original *Upward* tag = $b_{00}b_{01}b_1b_2 \ldots b_{n-1}$

The packet at stage $i-1$ meets a faulty link between stage $i-1$ and stage i or the switch at stage i is faulty.

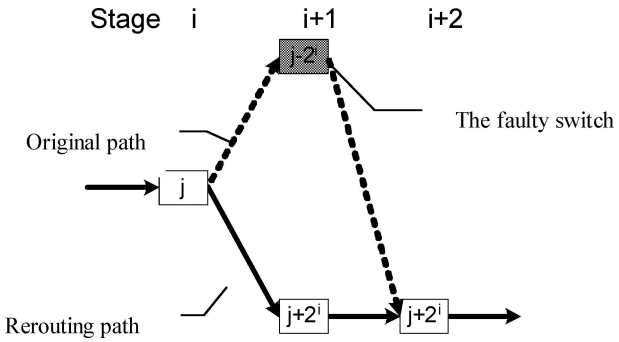

Figure 5. The pre-stage's switch j of the faulty switch successfully reroutes the packet that was originally sent to the faulty switch. The gray switch means the faulty switch.

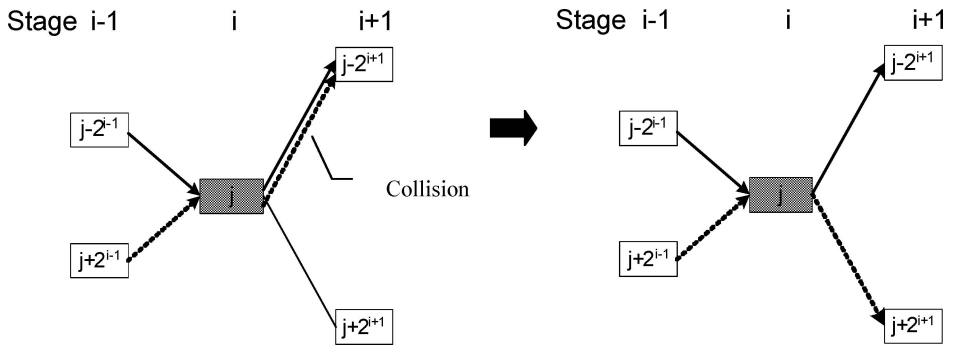

Figure 6. Two packets collide at switch j and be rerouted successfully by switch j at stage i.

Begin
  If ($i = 1$)
    {If ($c_{00}c_{01}/b_{00}b_{01} = 10/00$)
      {$c_{00}c_{01}/b_{00}b_{01} = 00/10$; the remaining routing tag $= b_1b_2 \ldots b_{n-1}b_n/c_1c_2 \ldots c_{n-1}c_n$}
    Else If ($c_{00}c_{01}/b_{00}b_{01} = 11/01$)
      {$c_{00}c_{01}/b_{00}b_{01} = 01/11$; the remaining routing tag $= b_1b_2 \ldots b_{n-1}b_n/c_1c_2 \ldots c_{n-1}c_n$ }
    }
  Else
   {
    If ($c_i/b_i = 1$)
      {$c_i/b_i = \overline{1}$; the remaining routing tag $= b_{i+1} \ldots b_{n-1}b_n/\, c_{i+1} \ldots c_{n-1}c_n$}
    Else If ($c_i/b_i = \overline{1}$)
      {$c_i/b_i = 1$; the remaining routing tag $= b_{i+1} \ldots b_{n-1}b_n/\, c_{i+1} \ldots c_{n-1}c_n$ }
    Else if (c_i/b_i=0)
      {
       $c_i/b_i = b_i/c_i$;
       Packet via the backward straight-link back to the switch of stage i
       as that of stage $i + 1$;
       The remaining routing tag $= b_ib_{i+1} \ldots b_{n-1}b_n/\, c_ic_{i+1} \ldots c_{n-1}c_n$;
       }
   }
  Output the rerouting tag
End

3.3.2.1. Rerouting method while traversing a non-straight link. In this section, we present the rerouting situations at the switch at stage i for the following two conditions: (1). A packet is sent to a non-straight output faulty link or to a faulty switch at stage $i+1$ by a non-straight link. (2). Two packets are delivered to a non-straight output link simultaneously. In these cases, the rerouting action occurs at the switch at stage i. In the case of meeting a faulty element, the switch at stage i computes the rerouting tag and sends the packet to the other non-straight link. In the collision case, the switch at stage i sends one of two collision packets to the other non-straight link to solve the collision. However, if there are three packets routed to the same non-straight link, the rerouting action fails. In the previous two cases, the routing tag is changed from *Downward* to *Upward* or from *Upward* to *Downward* for further routing. After the rerouting behavior, the packet is on the other disjoint path. Example 2 illustrates the rerouting situation.

Example 2 The source is 2, the destination is 4, and the switch 0 at stage 2 is faulty (or the upward link to this switch is faulty), as shown in Figure 7. The routing and rerouting paths are described as follows:

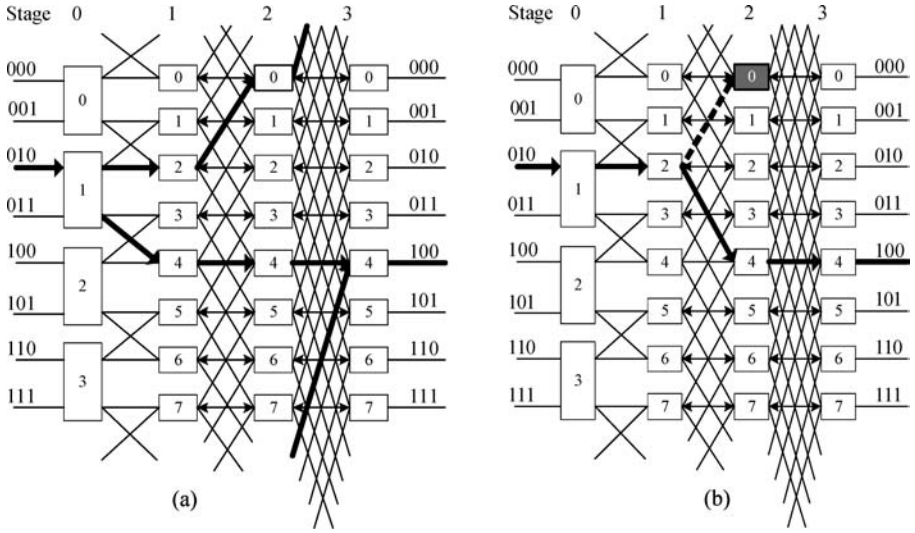

Figure 7. The rerouting condition in CSMIN with source $S = 2$ and target $T = 4$. Figure (a) shows the two disjoint paths, (1, 2, 0, 4) and (1, 4, 4, 4) where the 4-tuple item mean the switch indices at stage 0, 1, 2, and 3 respectively. In Figure (b), switch 2 at stage 1 reroutes a packet to switch 4 at stage 2 that is in the other disjoint path. As a result, the packet can be delivered along the other disjoint path to tolerate the faulty switch that is indicated by gray color and the dash line means the original routing path.

Solution:

The packet encounters a faulty element from switch 2 at stage 1 to switch 0 at stage 2. Therefore, switch 2 at stage 1 reroutes the packet. We assume that the packet uses the *Upward* tag.

The original path from stage 0 to stage 1 is described as follows:

Using the *Upward* tag: 2 (stage 0) $\rightarrow$ 2 (stage 1) $\rightarrow$ (x) 0 (stage 2) $\rightarrow$ 4 (stage 3), where (x) indicates the location of the faulty element.

Switch 2 at stage 1 computes the rerouting tag by using Algorithm 2 and reroutes the packet to the downward non-straight link (the other non-straight link). The packet uses the *Downward* tag to its destination. The number of rerouting hops to find the other disjoint path is 0.

1. At stage 1, the packet is sent to downward non-straight-link to stage 2: 2(stage 0) $\rightarrow$ 2(stage 1) $\rightarrow$ 4(stage 2).
2. At stage 2, switch 4 uses the *Downward* tag to route the packet to the destination: 2 (stage 0) $\rightarrow$ 2 (stage 1) $\rightarrow$ 4 (stage 2) $\rightarrow$ 4 (stage 3).

3.3.2.2. Rerouting method while traversing a straight link. In this section, we discuss the rerouting condition that is arisen from traversing a straight link. Switch j at stage i reroutes packets for avoiding meeting the faulty element or for preventing a collision in the following

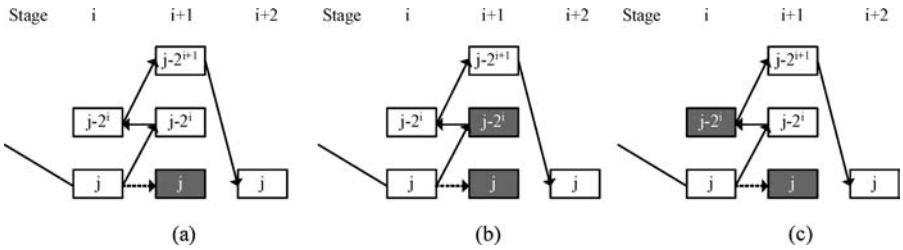

Figure 8. Switch j at stage i wants to route a packet to the switch j at stage i and the packet uses the *Downward* tag. In Figure (a), when switch j at stage $i+1$ is faulty, switch j reroutes the packet to switch $j-2^i$. In Figure (b), when switch j and $j-2^i$ both at stage $i+1$ are faulty, switch j cannot reroute the packet. In Figure (c), when switch j at stage $i+1$ and $j-2^i$ at stage i are faulty, the rerouted packet in the switch $j-2^i$ at stage $i+1$ cannot reroute the packet because switch $j-2^i$ at stage i is faulty. The dash line means the original path and the gray blocks mean faulty switches.

conditions: (1). A packet at stage i meets a faulty switch at stage $i+1$ via a straight-link or is delivered to a faulty straight link to stage $i+1$. (2). Two more packets at stage i want to pass through the same straight ink simultaneously.

The rerouting behavior can be described in the following two steps. In the first step, switch j at stage i reroutes the packet to switch $j+2^i$ at stage $i+1$ via the downward non-straight link if the packet uses the *Upward* tag. In contrast, if the packet uses the *Downward* tag, switch j reroutes the packet to switch $j-2^i$ via the upward non-straight link as shown in Figure 8(a). Besides, the switch j at stage i also changes the routing tag from the *Upward* tag to the *Downward* tag or from the *Downward* tag to the *Upward* tag. The second step is that switch $j+2^i$ or switch $j-2^i$ at stage $i+1$ sends the packet back to stage i via the backward straight link. After traversing two hops, the packet is on the other disjoint path. In the second step, the switch at stage $i+1$ can easily decide to deliver the packet to the next stage or previous stage by checking the original i-th routing bit, that is, if the packet is sent to stage $i+1$ by the non-straight link and the original i-th routing bit is 0, the switch at stage $i+1$ delivers the packet to the backward straight link to stage i. Example 3 illustrates the rerouting condition. However, if switch j and switch $j-2^i$ ($j+2^i$) at stage $i+1$ are both faulty and the packet is using the *Downward* (*Upward*) tag, switch j cannot reroute the packet as shown in Figure 8(b). In addition, if switch j reroutes a packet to switch $j-2^i$ ($j+2^i$) at stage $i+1$ and switch $j-2^i$ ($j+2^i$) at stage i is faulty too, switch $j-2^i$ ($j+2^i$) at stage $i+1$ discards the packet as shown in Figure 8(c).

Example 3 Let the source be 2 and the target be 4. Also, let the straight link connecting switch 4 at stage 1 and switch 4 at stage 2 be faulty, as shown in Figure 9(a). The routing and rerouting situations are described as follows:

Solution:

A. The original path (using *Downward* tag) in CSMIN is: 2 (stage 0) → 4 (stage 1) → (x) 4 (stage 2) → 4 (stage 3), where (x) indicates the faulty element.

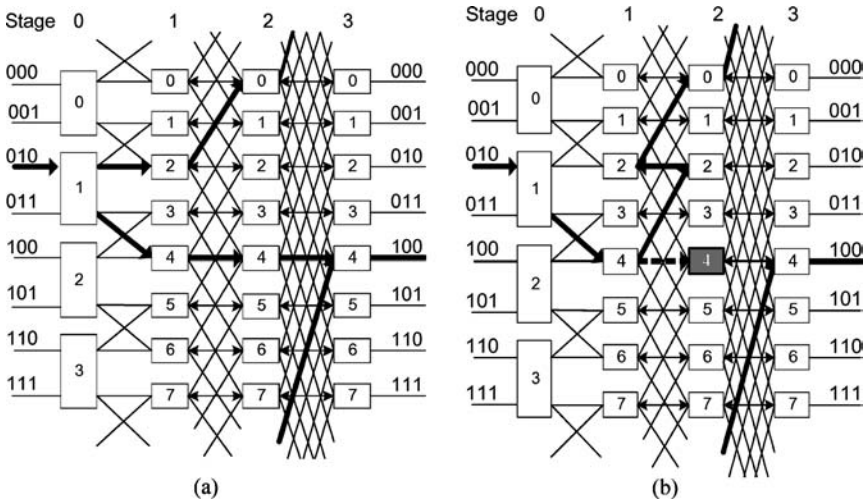

Figure 9. The rerouting condition in CSMIN with source $S = 2$ and target $T = 4$. Figure (a) shows the two disjoint paths, (1, 2, 0, 4) and (1, 4, 4, 4) where the 4-tuple item means the switch indices at stage 0, 1, 2, and 3 respectively. In Figure (b), switch 4 at stage 1 reroutes a packet to switch 2 at stage 2, and then switch 2 at stage 2 sends the packet back to switch 2 at stage 1. At this moment, the packet can be delivered along the other disjoint path to tolerate the faulty switch indicated by gray color and the dash line means the original routing path.

B. When a packet encounters a faulty element from stage 1 to stage 2, the switch at stage 1 reroutes the packet to the other disjoint path. The rerouting behavior is described as follows:

1. Because the *Downward* tag is used to route the packet originally, the switch at stage 1 takes the upward non-straight link to switch 2 at stage 2.: 2 (stage 0) $\rightarrow$ 4 (stage 1) $\rightarrow$ 2 (stage 2).
2. Because the i-th bit of Downward tag is 0, switch 2 at stage2 sends the packet to the backward straight link to switch 2 at stage 1.: 2 (stage 0) $\rightarrow$ 4 (stage 1) $\rightarrow$ 2 (stage 2) $\rightarrow$ 2 (stage 1).
3. At stage 1, switch 2 computes the routing tag by 2's complement computation to the *Upward* tag. Thus, the *Upward* tag is used from stage 1 to the destination if no more busy or faulty element is encountered: 2 (stage 0) $\rightarrow$ 4 (stage 1) $\rightarrow$ 2 (stage 2) $\rightarrow$ 2 (stage 1) [changing routing tag to the *Upward* tag] $\rightarrow$ 0 (stage 2) $\rightarrow$ 4 (stage 3).

4. Comparison and experimental results

In this section, we list the comparison and experimental results for five networks, CGIN, GIN, B-network, our proposed networks including CSMIN without backward straight links and CSMIN with backward straight links with fault-tolerant capacity and arrival ratio under variable traffic ratios.

Table 1. The comparison of our networks with GIN, CGIN, and B-Network

Network	Single fault tolerance	Crossbar architecture in the switches at middle stages	Fault-tolerant method
GIN	No	3×3	Dynamic rerouting
CGIN	Yes	3×3	Disjoint paths with two packets sent concurrently
B-network	No	3×3	Dynamic rerouting
CSMIN without backward straight links	Yes	3×3	Disjoint paths with two packets sent concurrently
CSMIN with backward straight links	Yes	4×4	Disjoint paths and Dynamic rerouting

4.1. Comparison

Table 1 shows the characteristics of some fault-tolerant networks including the crossbar architecture in the switches of middle stages, fault-tolerance capability, and the fault-tolerant method used by the network. These networks include GIN, CGIN, B-network, CSMIN without backward straight links and CSMIN with backward straight links.

In terms of hardware cost, CGIN and CSMIN without backward straight links costs 3×3 crossbar switches and provides two disjoint paths between any source-target pair to tolerant one fault. However, when the method of sending two identical packets along the two disjoint paths is used to increase the arrival ratio, these two networks result in more collisions and reduces the arrival ratio. With regard to GIN and B-network, these two networks use dynamic rerouting method to prevent collisions but they cannot guarantee one fault tolerance. B-network modified one of the redundant links in the GIN to be a backward link. Once a collision occurs, B-network sends a packet to the backward link to prevent the collision. Accordingly, B-network cannot neither prevent a collision occurring in the switches at stage 0 nor tolerate a faulty link between the first two stages. The capability of preventing collisions in the GIN is that a collision occurs in a straight output link; that is, one half of collisions in the GIN cannot be prevented. Although CSMIN with backward straight links costs 4×4 crossbar switch in hardware cost, it provides the capability of tolerating one fault and dynamic rerouting to prevent collisions. In next section, we simulate the five networks under without faults, with a switch fault, and with two switch faults to get the arrival rate, fault-tolerant capability.

4.2. Experimental results

In this section, we present our simulation model, results and discussions. We compare the throughput (arrival rate) with some fault-tolerant networks, including Gamma networks (GIN), CGIN, B-network, CSMIN without backward straight links, and CSMIN with backward straight links, under a traffic load of 6.25 to 100% with network size $N = 16, 32,$

and 64. CSMIN with backward straight links switch packets between two disjoint paths when a faulty or busy element is encountered while CSMIN without backward straight links sends two identical packets simultaneously to tolerate faults or solve the collision. In addition, we also simulate another network called CGIN, which also provides two disjoint paths to guarantee one fault-tolerant. In these two networks, CSMIN without backward straight links and CGIN, we send two identical packets concurrently along the two disjoint paths to get the arrival rate because they only provide two disjoint paths and no more extra hardware for switching packets between these two disjoint paths. With regard to computing a successful route, when one or both of the two identical packets arrives at the target, the route is successful.

According to the networks with none, one, or two separate faulty switches, we simulate the networks mentioned above to obtain the relevant results of arrival rate, collision rate, and fault-tolerant capability. We assume that the faulty switch is fully faulty, that is, the faulty switch cannot receive or send any packets from any input links or output links. In this case, the rerouting behavior occurs at the previous stage, whereas GIN, B-network, and CSMIN with backward links use this method and CGIN and CSMIN without backward straight links sends two identical packets concurrently to increase arrival ratio.

Figure 10 shows the arrival rates and the collision rates of these networks without faults when the network size could be 16, 32, and 64. CSMIN with backward straight link has better throughput than other networks no matter what the traffic loads are, and keeps lower collision rates all along, because it can reroute the collision packets in each stage with only one rerouting hop on average. With regard to CGIN and CSMIN without backward straight links, because the method of sending two identical packets two disjoint paths at the same

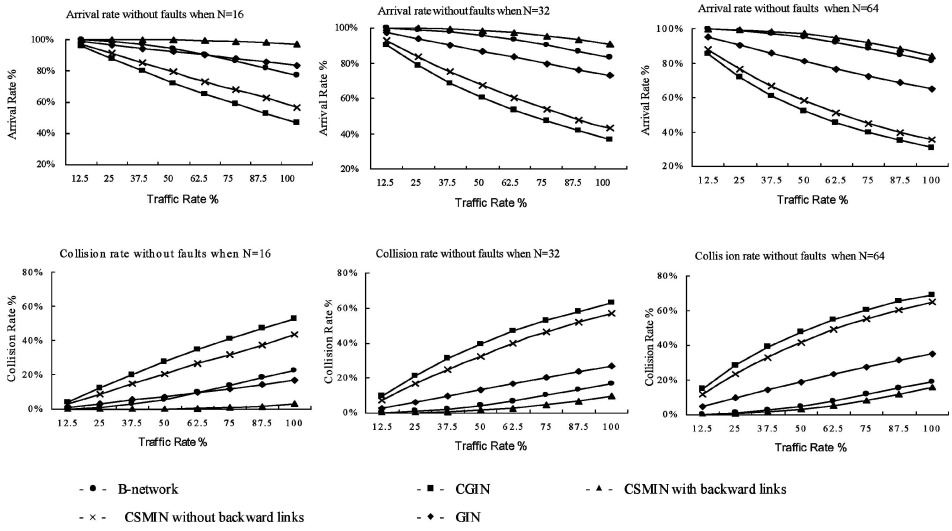

Figure 10. The arrival rates and collision rates of B-network, CGIN, CSMIN with/without backward straight links, and GIN without faulty switches when the network sizes are 16, 32, and 64 separately.

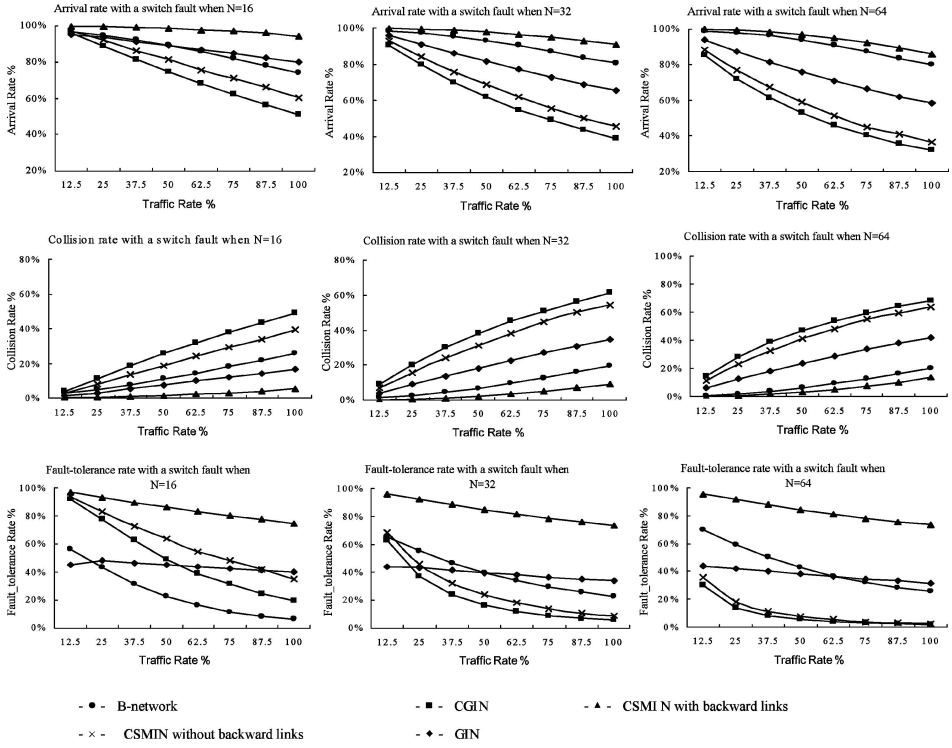

Figure 11. The arrival rates, collision rates and fault-tolerant rates of B-network, CGIN, CSMIN with/without backward straight links, and GIN without faulty switches when the network sizes are 16, 32, and 64 separately.

time results in packets collisions, the arrival rates of these two networks perform worse arrival rates.

Figures 11 and 12 present the arrival rates, collision rates and fault-tolerant rates of the networks when the situations with a fully faulty switch and two fully faulty switches are considered. In low traffic, CGIN and CSMIN without backward straight links, which send two identical packets via two disjoint paths at the same time perform better arrival ratio than or almost equal arrival ratio to GIN and B-network because these two networks can tolerate a faulty switch. However, the method of sending two identical packets causes rapid performance degradation because of packet collisions. Therefore, CGIN and CSMIN without a backward straight link present worse arrival ratio than other networks when the traffic load is high. In addition, Figures 11 and 12 also present the fault-tolerance capability of these networks. CGIN and CSMIN without backward straight links have higher fault-tolerant ratio than GIN and B-network, but the collision losses is greater than the benefits from tolerating faults. However, CSMIN with backward straight links that can tolerate faults and prevent collisions, so it performs better arrival ratio than other networks.

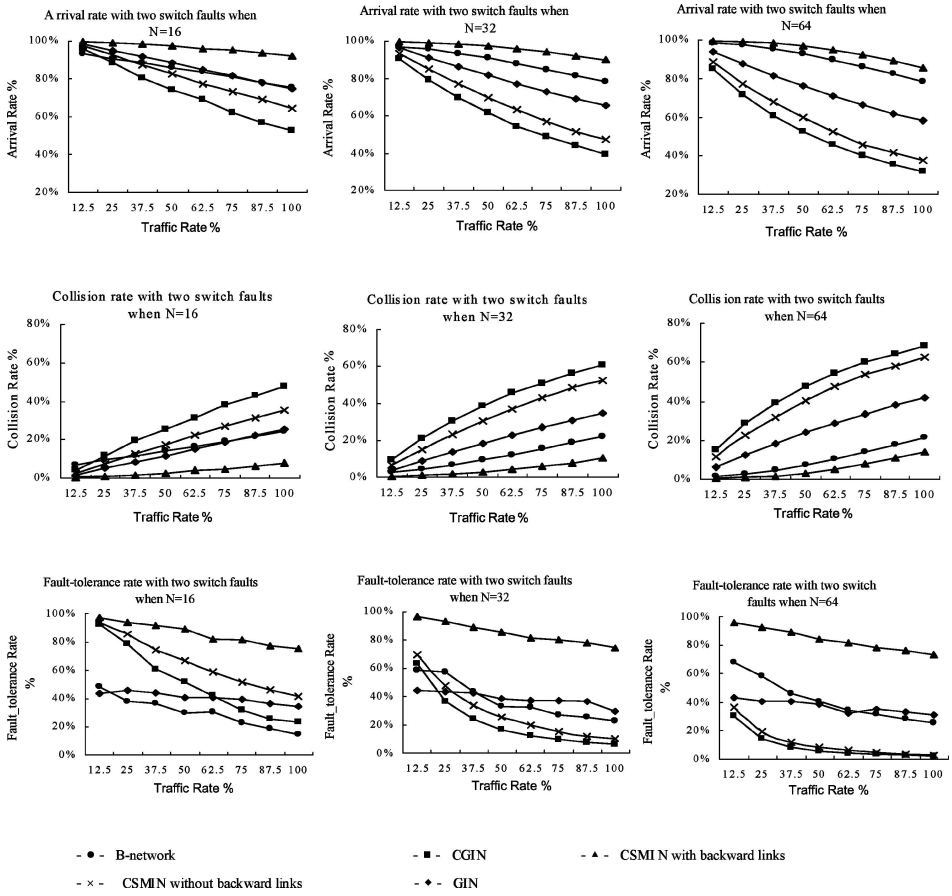

Figure 12. The arrival rates, collision rates, and fault-tolerant rates of B-network, CGIN, CSMIN with/without backward straight links, and GIN with two faulty switches when the network sizes are 16, 32, and 64 separately.

5. Conclusion

When a fault-tolerant network using disjoint paths is designed, the following issues should be considered: (1). fault-tolerant guarantee. (2). dynamic rerouting between disjoint paths. (3). rerouting hops; and (4). collision rate (arrival rate). In this paper, we presented a fault-tolerant network CSMIN with two disjoint paths to guarantee one fault-tolerance. Besides, because the vertical distance of these two disjoint paths at stage i is 2^i, the switch can dynamically reroute a packet from one disjoint path to the other by adding one backward straight link to CSMIN. Because CSMIN has the ability to switch packets between these two routing paths at each stage and keeps low rerouting hops (one rerouting hop on average), the collision ratio of CSMIN is lower than GIN, B-network and CGIN if CGIN sends two identical packets to tolerate a faulty element or solve the collision. From the simulation results, CSMIN performs well, with a better arrival ratio than other networks.

The virtues of CSMIN are the direct results of a good design philosophy: CSMIN is devised with two disjoint paths that a packet can easily be switched at each stage. This property is essential for efficient rerouting since, if disjoint paths in such a network have no the property at different stages, the rerouting very likely will have to return a packet by backtracking all the way back to the source, and then take the other disjoint path to the destination or sends two identical packets to tolerate a fault or solve a collision. However, such designs increase hardware cost and collision ratio.

With proper topology design, a fast strong rerouting scheme is achieved at different stages. As a result, CSMIN has advantages over other related disjoint paths networks in both re-routability and fault-tolerance capability.

References

1. G. B. III Adams, D. P. Agrawal, and H. J. Siegel. A survey and comparison of fault–tolerant multistage interconnection networks. *IEEE Transactions on Computers*, 20(6):14–27, 1987.
2. C. W. Chen, N. P. Lu, T. F. Chen, and C. P. Chung. Fault–tolerant gamma interconnection networks by chaining. In *IEE Proceedings on Computers and Digital Techniques*, 147(2):75–80, 2000.
3. P. J. Chuang. CGIN: A fault tolerant modified gamma interconnection network. *IEEE Transactions on Parallel and Distributed Systems*, 7(12):1301–1306, 1996.
4. P. J. Chuang. Creating a highly reliable modified gamma interconnection network using a balance approach. In *IEE Proceedings of Computers and Digital Techniques*, 145(1):27–32, 1998.
5. F. C. M. Lau and W. C. Poon. Throughput analysis of B-networks. *IEEE Transaction on Computers*, 47(47):482–485, 1998.
6. K. Y. Lee and H. Yoon. The PM22I interconnection network. *IEEE Transactions on Computers*, 38(Issue 2):302–307, 1989.
7. K. Y. Lee and H. Yoon. The B-network: A multistage interconnection network with backward links. *IEEE Transactions on Computers*, 39(7):966–969, 1990.
8. R. J. McMillen and H. J. Siegel. Performance and fault tolerance improvements in the inverse augmented data manipulator network. In *9th Symp. Computer Architecture*, pp. 63–72, April 1982.
9. D. S. Parker and C. S. Raghavendra. The gamma network. *IEEE Transactions on Computers*, C–33:367–373, April 1984.
10. C. S. Raghavendra and D. S. Parker. Reliability analysis of an interconnection network. In *Proceedings of 4th International Conference on Distributed Computing Systems*, pp. 461–471, May 1984.
11. D. Rau, J. A. B. Fortes, and H. J. Siegel. Destination tag routing techniques based on a state model for the iadm network. *IEEE Transactions on Computers*, 41(3):274–285, 1992.
12. S. W. Seo and T. Y. Feng. The composite banyan network. *IEEE Transactions on Parallel and Distributed Systems*, 6(10):1043–1054, 1995.
13. K. Yoon and W. Hegazy. The extra stage gamma network. *IEEE Transactions on Computers*, 37(11):1445–1450, 1988.