J Supercomput (2011) 55: 400–431
DOI 10.1007/s11227-009-0336-z

Designing a Fault-tolerant Fully-Chained Combining Switches Multi-stage Interconnection Network with Disjoint Paths

Nitin · Shruti Garhwal · Neha Srivastava

Published online: 8 October 2009

Abstract Multi-stage Interconnection Networks (MINs) are designed to achieve fault-tolerance and collision solving by providing a set of disjoint paths. Ching-Wen Chen and Chung-Ping Chung had proposed a fault-tolerant network called Combining Switches Multi-stage Interconnection Network (CSMIN) and an inaccurate algorithm that provided two correct disjoint paths only for some source-destination pairs. This paper provides a more comprehensive and accurate algorithm that always generate correct routing-tags for two disjoint paths for every source-destination pair in the CSMIN. The 1-fault tolerant CSMIN causes the two disjoint paths to have regular distances at each stage. Moreover, our algorithm backtracks a packet to the previous stage and takes the other disjoint path in the event of a fault or a collision in the net-

Nitin, Member, SIAM, IEEE and ACM

Nitin (✉)
Department of Computer Science and Engineering and Information Technology, Jaypee University of Information and Technology, Waknaghat, Solan 173215, Himachal Pradesh, India
e-mail: delnitin@ufl.edu

Nitin
e-mail: delnitin@ieee.org

Nitin
e-mail: delnitin@gmail.com

Nitin
e-mail: delnitin@juit.ac.in

S. Garhwal
Accenture Services Private Limited, Building 1B, Raheja Mindspace, Madhapur 500081, Hyderabad, India
e-mail: shruti.garhwal@accenture.com

N. Srivastava
Accenture Services Private Limited, Tower 5, Cybercity, 143, Magarpatta City, Hadapsar–Mundhwa Road, Hadapsar 411013, Pune, India
e-mail: neha.srivastav@accenture.com

work. Furthermore, to eliminate the backtracking penalties of CSMIN, we propose a new design called Fault-tolerant Fully-Chained Combining Switches Multi-stage Interconnection Network (FCSMIN). It has similar characteristics of 1-fault tolerance and two disjoint paths between any source-destination pair, but it can tolerate only one link or switch fault at each stage without backtracking. Our simulation and comparative analysis result shows that FCSMIN has added advantages of destination-tag routing, lower hardware costs, strong reroutability, lower preprocessing overhead, and higher fault-tolerance power in comparison to CSMIN.

Keywords Multi-stage Interconnection Network · Combining Switches Multi-stage Interconnection Network · Fault-tolerant Fully-Chained Combining Switches Multi-stage Interconnection Network · Collision solving · Routing-tag Algorithm · Rerouting tag · Distance-tag algorithm and Disjoint Paths

1 Introduction and motivation

Interconnection Networks (IN) [1–10] are used to design a network in which there are several independent paths between two modules being connected which increases the available bandwidth. Many stages of inter-connected switches form a MIN. For high reliability and performance, several methods have been suggested that provide fault-tolerance to MINs [11–18]. The basic idea of in case of fault-tolerance is to provide multiple paths for a source-destination pair, so that the alternate paths can be used in case of a fault in the path. However, to guarantee 1-fault tolerance, a network should have a pair of alternate paths for every source-destination pair which are disjoint in nature [1–8].

Previous work in this direction by Ching-Wen Chen and Chung-Ping Chung in [19] proposed a fault-tolerant network called CSMIN and an incorrect algorithm that did not provide two correct disjoint paths for some source-destination pairs. Their work did not generate correct routing-tags for some source-destination pairs. The routing-tags generated for such source-destination pairs were not correct in the sense that the resulting two disjoint paths in CSMIN did not reach the desired destination. This paper provides a more comprehensive and accurate algorithm that always generates correct routing-tags for every source-destination pair in the CSMIN so that the resulting two disjoint paths reach the desired destination.

Our algorithm can also dynamically reroute packets between these two paths to solve the faults or collision situation for every source-destination pair in CSMIN.

With the aim to achieve the demands of high reliability, many prior researchers have worked upon the objective of making MINs fault-tolerant. The fault-tolerance capability in a MINs guarantees that a packet will have an alternative routing path if it encounters a faulty or busy switch or a communication link in its original routing path [1–8]. A MIN is able to meet the reliability demands if it is at least 1-fault tolerant, i.e. there is at least one alternative path to deal with faults or collisions. This alternative path should be disjoint with the original routing path followed and it would not have any implication whenever a switch or a link fails in the original routing path (then the alternative path will also fail). Most of the MINs do not generate two disjoint paths,

are not fault-tolerant, and hence (in turn) will result in packet losses and eventually degradation in the performance. Moreover, to improve this situation, we used to have two disjoint paths, which always guarantees a solution of the problem of faults or collisions in a network.

Furthermore, we propose a new design called Fault-tolerant Fully-Chained Combining Switches Multi-stage Interconnection Network (FCSMIN) that makes use of destination-tag routing for stages 1 to n to overcome the backtracking problem in CSMIN. FCSMIN has the similar characteristics of 1-fault tolerant and two disjoint paths between any source-destination pair. However, it can tolerate only one link or switch fault at each stage without backtracking. For stages 1 to n, chaining links are added between nodes that belong to a neighboring group at the same stage. When a link fault occurs at a stage in FCSMIN, the chaining link is used. We also introduce two new destination-tag routing functions, UpRoute and DownRoute, which can be used to find two disjoint paths in FCSMIN. One can find the literature regarding the destination-tag algorithm and dynamic rerouting in [19–22].

The rest of the paper is as follows. Section 2 explains the basics of MINs, fault-tolerance, and disjoint-path networks. Section 3 provides an insight into the topology and the salient features of the 1-fault tolerant CSMIN. It also covers our proposed accurate algorithms that provide two disjoint paths for every source-destination pair and the dynamic rerouting between the two disjoint paths to solve collisions or faults for every packet. Section 4 provides the details of comparison, experimental setup, and simulation results of our algorithm in terms of arrival and collision ratio for every source-destination pair in CSMIN. Section 5 covers the proposed design known as FCSMIN with chaining links and with multiplexers and demultiplexers. The FCSMIN uses a dynamic algorithm for routing and easy rerouting using either the UpRoute or DownRoute function. Section 6 presents a comparative analysis of FCSMIN over CSMIN followed by the conclusion and future scope provided in Sects. 7 and 8, respectively.

2 Preliminaries and background

2.1 Multi-stage interconnection networks

MINs are currently used for many different applications, ranging from internal buses in Very Large-Scale Integration (VLSI) circuits to wide area computer networks. It connects input devices to output devices through a number of switch stages, where each switch is a crossbar network. The number of stages and the connection patterns between stages determine the routing capability of the networks. The lack of standards and the need for very high performance and reliability pushed the development of MIN for parallel computers with hundreds of processors and some commercial machines. Since the assurance of high reliability is a significant task in complex systems, fault-tolerance is crucial for MINs to serve the communication needs. In the absence of faults, the most important performance metrics of a MIN are system latency and throughput.

2.2 Fault-tolerance aspects of MINs

The fault-tolerance capability in a MIN guarantees that a packet will have an alternative routing path if it encounters a faulty or busy switch or a communication link in its existing routing path. A MIN will entirely meet the reliability demands if it is at least 1-fault tolerant, i.e. there is at least one alternative path to deal with faults or collisions. This alternative path should be disjoint in nature with the existing routing path followed. The performance of a MIN in terms of its throughput is highly dependent on its collision solving ability. A MIN should be to reroute packets on an alternative path when two or more packets are in conflict for the use of a resource such as a switching element or a communication link in the existing routing path. The lesser the number of packets lost due to collision the better is its efficiency in solving collisions; The better the collision solving ability, the better the performance.

With the aim to achieve the above objectives of fault-tolerance and collision solving, we attempt to design and simulate a MIN that is at least 1-fault tolerant and has a high rate of collision solving. Many prior researches and developments have been made in this direction. Many designs and routing algorithms for MINs have been put forth to effectively deal with faults and collisions in the network. The nature of these designs and algorithms have been characterized to either compromise, balance or optimize, all, any, or some of the following factors such as cost-effectiveness, reliability, throughput, communication delay, pre-processing overhead, and memory capacity. Our work was inspired by the existing approaches that led to the design of several regular, irregular, and hybrid MINs. These approaches exploited the topology of a MIN in the following ways:

1. Different number of switching elements at each stage.
2. Adding or removing extra stages.
3. Changing the nature of communication links from straight to non-straight upward or downward.
4. Introducing buffer in the switching elements.
5. Introducing a centralized controller in the form of additional circuitry for the control logic.
6. Introducing chaining links in some or all stages.
7. Introducing multiplexers and demultiplexers in stages 0 and n.
8. Combining the topologies of two or more MINs.

Many significant changes have been made in the routing schemes adopted for MINs with aim of minimizing latency, easy rerouting, and a decrease in pre-processing overhead. Previous approaches or solutions were mostly blocking in nature. They always resulted in high rates of packet losses due to collisions or faults. Some regular networks like the cube interconnection network [2] provided only one path for routing packets between any source and destination node. If this path failed, no other path existed to route the packets, and hence the packet was lost resulting in performance degradation. Some irregular networks like Double Order Tree Interconnection Network (DoT) [11] provided more than one path of different path lengths for some source-destination pairs. This one path had only one switching element in its middlemost stage and whose failure could result in a choking condition. There were

also other approaches, explained in [9] as the hybrid ZETA Network, Augmented Baseline Network, Quad-tree Network, and Augmented Shuffle-Exchange Network, which uses multiplexers, demultiplexers, and chaining links in an attempt to provide fault-tolerance. However, these approaches were only fault-tolerant for some cases. Then there were some MINs like Gamma Interconnection Networks (GIN) [23], which were 1-fault tolerant. Although GIN provided two sets of paths to deal with a faulty or busy switch or link, these paths were not disjoint in nature 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. Furthermore, Gamma networks have only one single path when the indices of the source and the target are the same.

To address the problems of both performance and fault-tolerant capability, one can approach to design and simulate a 1-fault tolerant network with the following issues explained in [19] as:

1. Guarantee of at least two disjoint paths.
2. Easy rerouting between disjoint paths.
3. Keep low rerouting hops.
4. Solve the occurrences of packet collisions.

2.3 Previous work on providing disjoint paths

There has been extensive research on disjoint paths to guarantee fault-tolerance [19–25]. For example, these networks include modified Gamma Interconnection Network (CGIN) [24], Composite Banyan [26], Gamma Interconnection Network by chaining (PCGIN) [25], and Balanced Gamma Interconnection Network (BGIN) [25]. The BGIN and the composite banyan modified the redundant link to a symmetric link to provide two disjoint paths between any source target pair. In contrast with providing disjoint paths, B-Network [27, 28], which modified the GIN, provides the capability of dynamic rerouting to prevent the collisions during the routing path. However, B-Network cannot guarantee 1-fault tolerance. With regard to CGIN, 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 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 with the two disjoint paths. This method causes more packet collisions. The other method uses backtracking rerouting [26]. The backtracking is a method in which a switch is used 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 changed to bi-directional and the rerouting hops count is high. This causes an increase in the hardware cost and collision rate. Methods using extra stages to tolerate faults suffers from increased hardware cost and collision rate because no matter whether packets encounter a faulty or busy element or not, the length of routing paths still increases. Most of the text considered here is taken from [19].

In our paper, to guarantee 1-fault tolerance, easy rerouting capability between disjoint paths with low rerouting hops, we have considered the fault-tolerant network called Combining Switches Multistage Interconnection Network (CSMIN) [19] and implemented the above issues. Though this design proposed by Ching-Wen Chen and Chung-Ping Chung made an impressive attempt in meeting the above requirements, it had an inaccurate routing algorithm, which did not always generate correct disjoint paths for every source-destination pair. We have smartly modified the algorithm and provided a more accurate and comprehensive approach that always generates correct routing-tags for the two disjoint paths for every source-destination pair in the CSMIN. The 1-fault tolerant CSMIN causes the two disjoint paths to have regular distances at each stage. Moreover, our algorithm backtracks a packet to the previous stage and takes the other disjoint path in the event of a fault or a collision in the network. Thus, our approach guarantees 1-fault tolerance for all cases by providing two disjoint paths. Moreover, an easy rerouting algorithm has been proposed that simply put the packets on other disjoint paths to deal with faults or collisions. This rerouting algorithm makes use of the backtracking mechanism with a very low average rerouting hop of one. This backtracking mechanism not only increases the system latency, but also increases the hardware cost as bi-directional switches are used in CSMIN for stages 1 to n to achieve reroutability. In search for solutions to the above problems, we proposed a new design called FCSMIN.

FCSMIN has (the similar) characteristics of 1-fault tolerance and two disjoint paths between any source-destination pair, however, it can tolerate at least one link or switch fault at each stage without backtracking by making use of destination-tag routing algorithm for stages 1 to n. For stages 1 to n, chaining links are added between nodes, belonging to a neighboring group at the same stage. Whenever a link fault occurs at a stage in FCSMIN, a chaining link will be chosen automatically.

In the next section, we have provided the accurate routing-tag and distance-tag algorithms and strategic design issues of CSMIN and compared the same with FCSMIN, B-Network, GIN, and CGIN are considered as running examples throughout the paper.

3 CSMIN: accurate routing-tag and distance-tag algorithms and strategic design issues

This section provides an accurate algorithm that provides two correct disjoint paths and also generate correct routing-tags for the two disjoint paths for every source-destination pair in the CSMIN. The 1-fault tolerant CSMIN causes the two disjoint paths to have regular distances at each stage. Moreover, our algorithm backtracks a packet to the previous stage and takes the other disjoint path in the event of a fault or a collision in the network.

3.1 Topology of CSMIN

A CSMIN [19] of size $N = 2^n$ consists of $n+1$ stages labeled from 0 to n. At stage 0, switch $2i$ and switch $2i+1$ are coupled into a 2×4 switch, for $= 0$ to $(\frac{N}{2} - 1)$. Stage

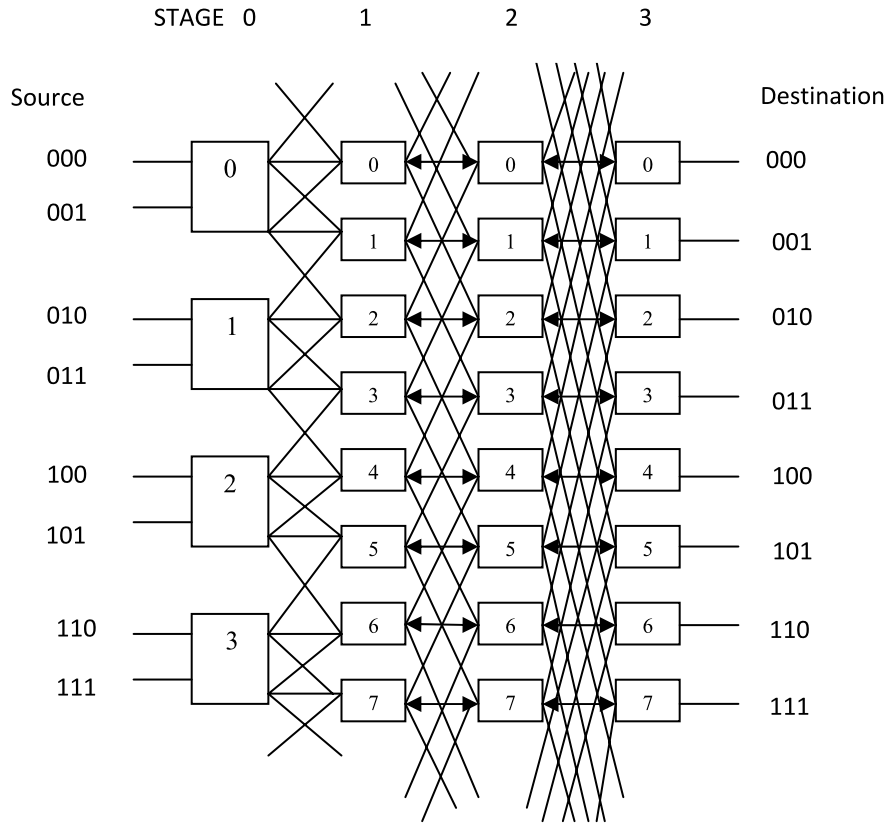


Fig. 1 Combining Switches Multi-stage Interconnection Network (CSMIN)

1 to Stage n have N switches labeled from 0 to $2^n - 1$. All straight links between stage 1 and stage n are bi-directional. The switch architecture at the first and the last stage has 2×4 and 3×2 crossbars, respectively. Switches located at stage 1 have 3×3 crossbars. Moreover, each switch located at the intermediate stage has a 4×4 crossbar switch. Figure 1 illustrates a CSMIN of size 8.

3.2 Routing-tag algorithm

The previous work on CSMIN had used routing-tags to identify the two disjoint paths for a source-destination pair. However, the earlier algorithms generated incorrect routing-tags for the two disjoint paths of some source-destination pairs. Our proposed algorithm provides comprehensive and accurate routing-tags for all source-destination pairs, so that CSMIN is completely 1-fault tolerant.

Our algorithm exploits the topology of CSMIN so that there exists two paths between any source-destination pair always having a regular vertical distance of 2^i at each stage i, where $1 \leq i \leq n - 1$, so that the two paths are always disjoint in nature. We define two distance-tag routing functions known as Downward and Upward. The

path generated 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 link.

From the routing-tags generated by the proposed algorithm, the first two routing-tag bits are used for transferring packets from stage 0 to stage 1. The four links to stage 1 from a 2×4 switch in stage 0 are downward non-straight, downward straight, upward straight, upward non-straight as shown in the CSMIN Fig. 1. These links are represented by the two bit combinations (00, 01, 10, 11), respectively. We describe the correct and accurate algorithm to generate the Downward and Upward routing-tags, which consist of $n + 1$ bits each for a CSMIN of size 2^n $(= N)$, in Algorithm 1.

3.3 Routing in CSMIN

In this section, we describe the routing behavior of CSMIN. 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, 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 $S = 5$ and the destination $T = 5$ with size $N = 8$.

Example 1 If source $S = 5$, destination $T = 5$, and network size $N = 8$, the routing situation in CSMIN is as shown in Fig. 2. The routing paths are described as follows:
Solution: In the following *Algorithm 1* for $S = 5$ and $T = 5$, we have

1. $S1 = 5$ and $T1 = 5$.
2. $S = 4$ since $T - S$ is even and S is odd.
3. DownwardD $= (5 - 4) \text{mod} 8 = 1 = 0001$ and UpwardD $= (8 - (5 - 4)) \text{mod} 8 = 7 = 0111 = 0222$.
4. DownwardD $= 1101$ and UpwardD $= 0122$ since $c_0 = 0$ and S is even.
5. DownwardD $= 1001$ and UpwardD $= 0022$ since $c_0 = 1$ and S is even.
6. DownwardD $= 1000$ since $S1 = 5$ and $T1 = 5$.

The Downward and Upward routing-tags thus generated from Algorithm 1 are 1000 and 0022, respectively. Take one of these as the main routing-tag.

3.4 Algorithm for dynamic rerouting

In this section, we introduce the rerouting methods when a faulty switch or link or a busy switch or link is encountered. We have used the backtracking scheme in which the switch sends the packet back along the traversed path to the pre-stage switch. The pre-stage switch takes the other disjoint path to tolerate the faulty or busy element. To switch packets (dynamically) between the two disjoint paths, the reversed straight link of the bi-directional straight link is used to reroute packets to the previously traversed switch in the previous stage of CSMIN.

We describe the algorithm for generating the rerouting-tags in Algorithm 2. If the packet encounters a faulty or busy element and if the main routing-tag followed was Downward, then the packet now follows the Upward routing-tag or vice versa.

Algorithm 1 Generating the routing-tag of two disjoint paths in CSMIN

1. Input the source S and target T
2. Let $S1 = S$ and $T1 = T$
3. **If** ($T - S$ is even)
   If (S is even)
   $S = S + 1$
   Else
   $S = S - 1$
   End If
4. Let distance DownwardD $= (T - S)\text{Mod}N$ and UpwardD $= (N - (T - S))\text{Mod}N$
5. Convert DownwardD to its binary representation $c_0c_1c_2 \ldots c_{n-1}$
6. Convert UpwardD to its binary representation $b_0b_1b_2 \ldots b_{n-1}$ and replace any 1 with 2
7. **If** ($c_0 = 0$ and S is odd)
   Replace c_0c_1 with 10 and b_0b_1 with 00
   Else If ($c_0 = 0$ and S is even)
   Replace c_0c_1 with 11 and b_0b_1 with 01
   End If
8. **If** ($c_0 = 1$ and S is odd)
   Replace c_0c_1 with 11 and b_0b_1 with 01
   Else If ($c_0 = 1$ and S is even)
   Replace c_0c_1 with 10 and b_0b_1 with 00
   End If
11. For Downward tag $c_0c_1c_2 \ldots c_{n-1}$ & For Upward tag $b_0b_1b_2 \ldots b_{n-1}$
   If ($s1 = 0/1$)
   If ($t1 = 1/2/3/4$)
   If ($c_{n-1} = 1$)
   $c_{n-1} = 0$
   Else If ($t1 = 0/5/6/7$)
   If ($b_{n-1} = 2$)
   $b_{n-1} = 0$
   End If
   If ($s1 = 2/3$)
   If ($t1 = 0/1/2/7$)
   If ($b_{n-1} = 2$)
   $b_{n-1} = 0$
   Else If ($t1 = 3/4/5/6$)
   If ($c_{n-1} = 1$)
   $c_{n-1} = 0$
   End If
   If ($s1 = 4/5$)
   If ($t1 = 1/2/3/4$)
   If ($b_{n-1} = 2$)
   $b_{n-1} = 0$
   Else If ($t1 = 0/5/6/7$)
   If ($c_{n-1} = 1$)
   $c_{n-1} = 0$
   End If
   If ($s1 = 6/7$)
   If ($t1 = 0/1/2/7$)
   If ($c_{n-1} = 1$)
   $c_{n-1} = 0$
   Else If ($t1 = 3/4/5/6$)
   If ($b_{n-1} = 2$)
   $b_{n-1} = 0$
   End If
12. Display Downward and Upward tag.

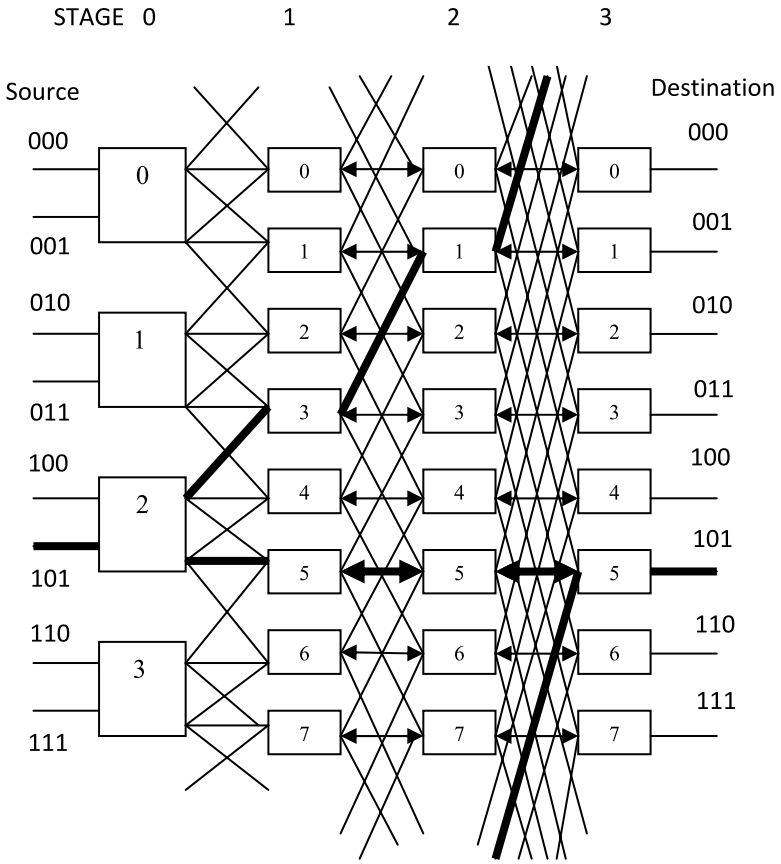


Fig. 2 The two disjoint paths $(2, 5, 5, 5)$ and $(2, 3, 1, 5)$ indicated by the *bold line* in CSMIN with source $= 5$ and target $= 5$, where the 4-tuple denotes the switch indices at stage $0, 1, 2$, and 3, respectively

Hence, the routing-tag is changed from Downward to Upward or from Upward to Downward for further routing. The switch traversed at the previous stage computes the rerouting-tag. After the rerouting behavior, the packet is sent on the other disjoint path.

3.4.1 Rerouting in CSMIN

In this section, we present two rerouting situations one while traversing a non-straight link illustrated by Example 2 and the other while traversing a straight link illustrated by Example 3.

Example 2 The source is 2, the destination is 3, and the switch 7 at stage 2 is faulty (or the upward link to this switch is faulty). The routing and rerouting paths are described as follows:
Solution: In the following *Algorithm 1* for $S = 2$ and $T = 3$, we have

Algorithm 2 Generating Rerouting-tag

The original Downward tag = $c_0c_1c_2 \ldots c_{n-1}$
The original Upward tag = $b_0b_1b_2 \ldots b_{n-1}$
The packet at stage $i-1$ meets a faulty switch at stage i
Begin
If $(i = 1)$
If $(c_0c_1/b_0b_1 = 10/00)$
$c_0c_1/b_0b_1 = 00/10$;
Remaining routing-tag = $b_2 \ldots b_{n-1}/c_2 \ldots c_{n-1}$
Else If $(c_0c_1/b_0b_1 = 11/01)$
$c_0c_1/b_0b_1 = 01/11$;
Remaining routing-tag = $b_2 \ldots b_{n-1}/c_2 \ldots c_{n-1}$
End If
Else If $(c_i/b_i = 1)$
$c_i/b_i = 2$;
Remaining routing-tag = $b_{i+1} \ldots b_{n-1}/c_{i+1} \ldots c_{n-1}$;
Else If $(c_i/b_i = 2)$
$c_i/b_i = 1$;
Remaining routing-tag = $b_{i+1} \ldots b_{n-1}/c_{i+1} \ldots c_{n-1}$;
Else If $(c_i/b_i = 0)$
$c_i/b_i = b_i/c_i$;
Remaining routing-tag = $b_ib_{i+1} \ldots b_{n-1}b/c_ic_{i+1} \ldots c_{n-1}$;
End If
Output the rerouting-tag
End

1. $S1 = 2$ and $T1 = 3$.
2. $S = 2$ since $T - S$ is odd.
3. DownwardD = $(3-2)\text{mod}8 = 1 = 0001$ and UpwardD = $(8-(3-2))\text{mod}8 = 7 = 0111 = 0222$.
4. DownwardD = 1101 and UpwardD = 0122 since $c_0 = 0$ and S is even.
5. DownwardD = 1001 and UpwardD = 0022 since $c_0 = 1$ and S is even.
6. DownwardD = 1000 since $S1 = 2$ and $T1 = 3$.

The Downward and Upward routing-tags thus generated from Algorithm 1 are 1000 and 0022, respectively. Take one of these as the main routing-tag. Figure 3a shows the two disjoint paths (1, 1, 7, 3) and (1, 3, 3, 3), where the 4-tuple means the switch indices at stage 0, 1, 2, and 3, respectively.

We assume that the packet uses the Upward tag. Since the switch 7 at stage 2 is faulty (or the upward link to this switch is faulty), the switch 1 at stage 1 computes the rerouting-tag by using Algorithm 2 and reroutes the packet on the downward non-straight link. The packet then uses the Downward tag to reach its destination. The number of rerouting hops to find the other disjoint path is 0. In Fig. 3b, switch 1 at stage 1 reroutes a packet to switch 3 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

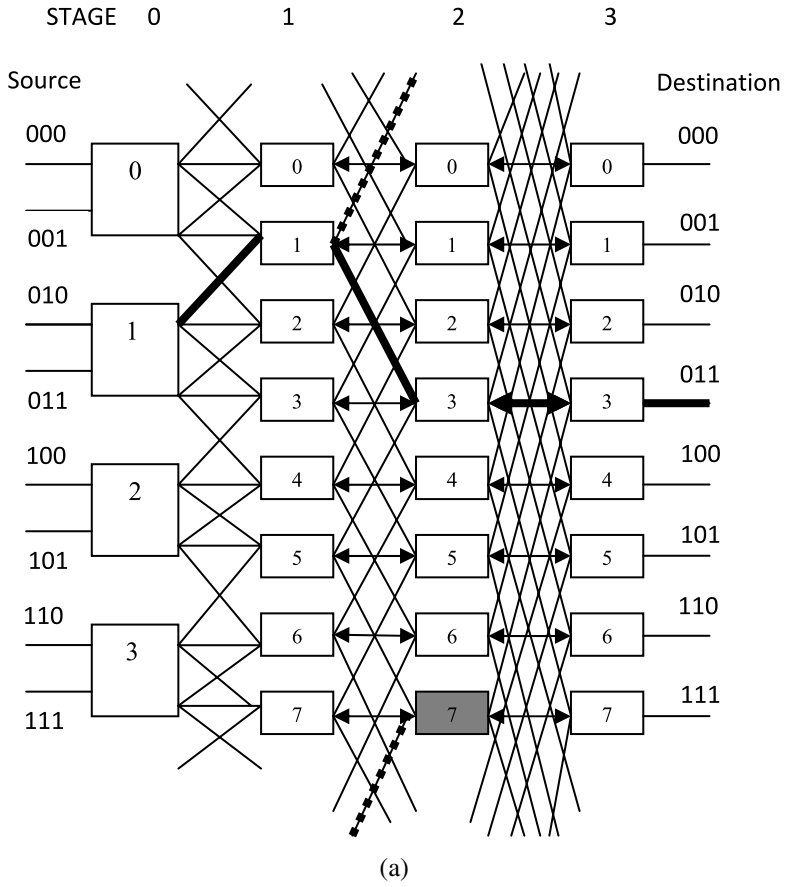
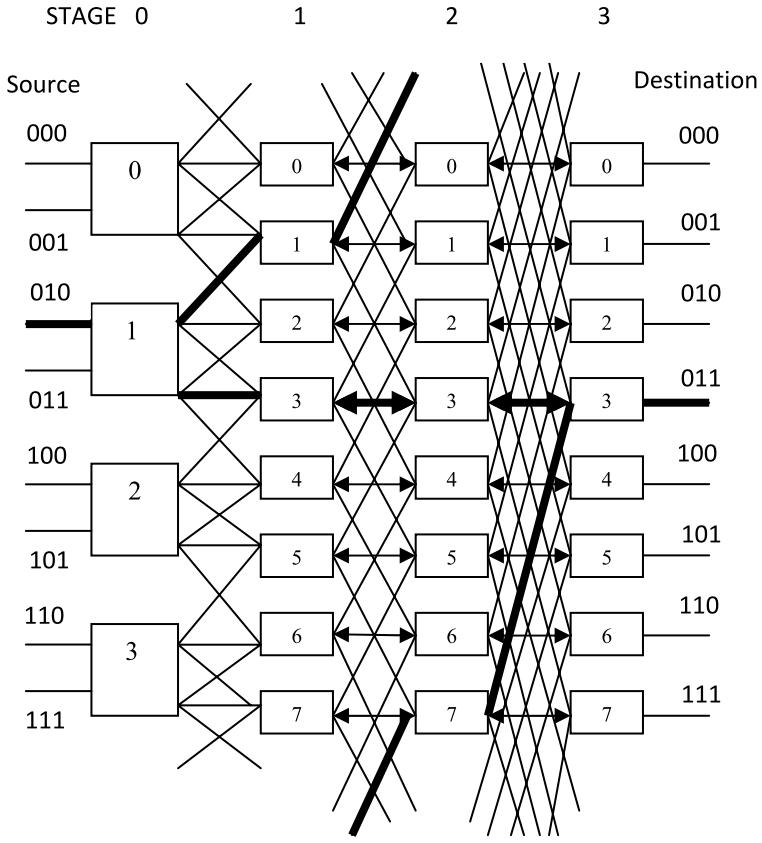


Fig. 3 The routing condition of CSMIN. (**a**) shows the routing condition in CSMIN with source $S = 2$ and $T = 3$ and (**b**) shows the rerouting condition in CSMIN with source $S = 2$ and $T = 3$ for Example 2. The packet can be delivered along the other disjoint path to tolerate the faulty switch indicated by *gray color* and the *dash line* means the original routing path

switch that is indicated by gray color and the dash line means the original routing path.

Example 3 The source is 2 and the destination is 3. The Downward and Upward routing-tags, thus generated from Algorithm 1 for this source-destination pair are 1000 and 0022, respectively. Take one of these as the main routing-tag. The routing and rerouting paths are described as follows.

Solution: Let the straight link connecting switch 3 at stage 1 to switch 3 at stage 2 to be faulty. When a packet encounters a faulty element from stage 1 to stage 2, the switch at stage 1 reroutes the packet on the other disjoint path as shown in Fig. 4. We assume that the packet uses the Downward tag. Since the Downward tag is used to route the packet originally, the switch at stage 1 takes the upward non-straight link to switch 1 at stage 2. Switch 1 at stage 2 then sends the packet on the backward straight



(b)

Fig. 3 (*Continued*)

link to switch 1 at stage 1. At Stage 1, switch 1 computes the Upward routing-tag, which in turn used to move from stage 1 to the destination (if no more busy or faulty element is encountered). The number of rerouting hops to find the other disjoint path is 1.

In the next section, we present the simulation of CSMIN with backward straight links, B-Network, CGIN, and GIN for network size of 16 and compare their arrival rate under a faulty-free situation with the situations with one fully faulty switch. Moreover, our result shows the improvements in the arrival ratio that results when a faulty switch is avoided after rerouting and compares that with the arrival ratio when a packet encounters a faulty switch again.

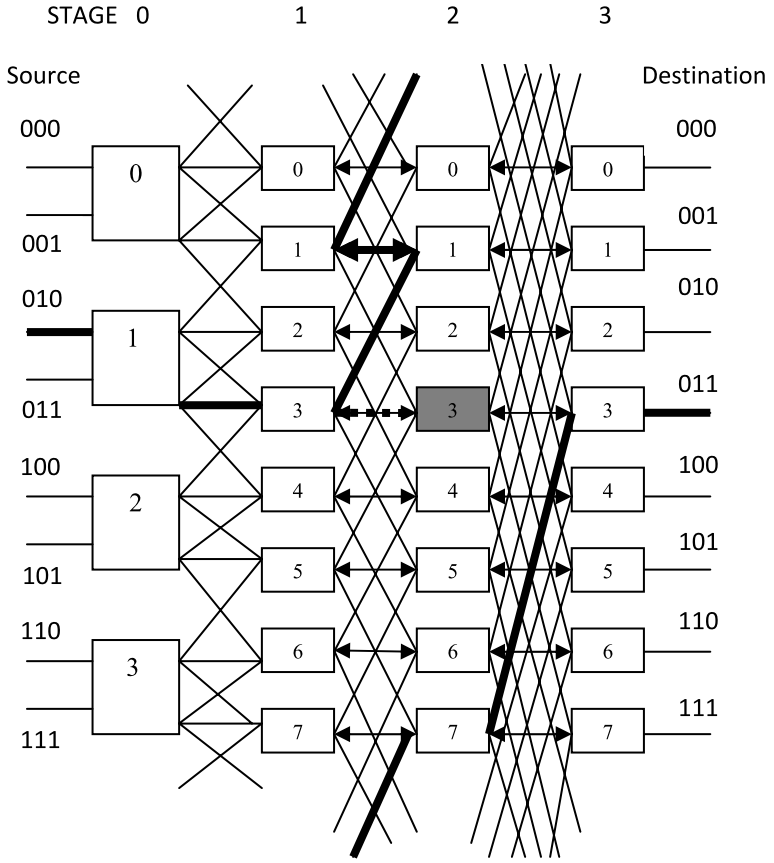
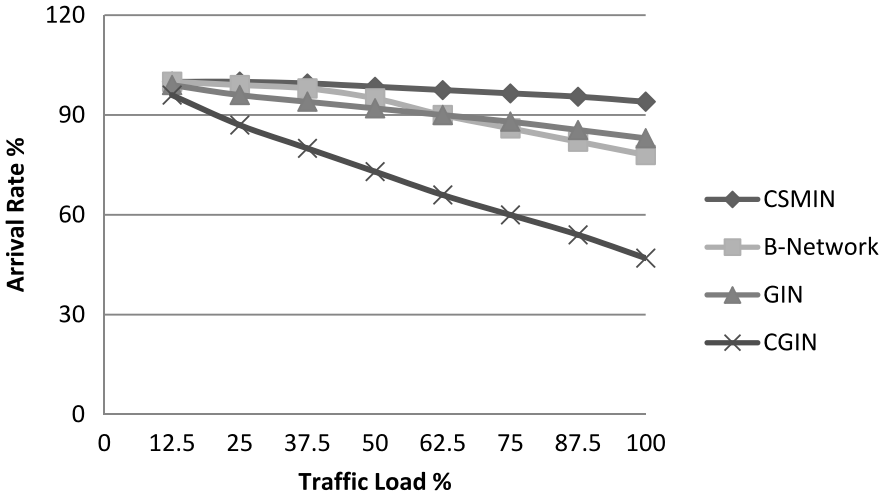


Fig. 4 The rerouting condition in CSMIN with source $S = 2$ and $T = 3$ for Example 3. The packet can be delivered along the other disjoint path to tolerate the faulty switch that is indicated by a *gray color* and the *dashed line* means the original routing path

4 Experimental setup and comparative analysis of CSMIN, B-Network, GIN, and CGIN based on simulation results

In this section, we present our simulation, results, and discussions. The previous work on the routing algorithm for CSMIN did not generate correct routing-tags for some source-destination pairs. The routing-tags generated for these source-destination pairs were not correct in the sense that the resulting two disjoint paths in CSMIN for the desired destination did not reach the desired destination. Since their routing algorithm was not valid, its simulation resulted in packets arriving at invalid destinations. Therefore, we have not considered their algorithm for comparison with ours.

For our simulation, we have used the IBM System x, running with Novell's SUSE Linux Enterprise Server 11, and we continuously and randomly generated the different source-destination requests in each cycle and continuously ran 32,768 cycles to compute the arrival rate, the collision rate, and fault-tolerance rate. The following cases have been addressed while computing the fault-tolerance rate:



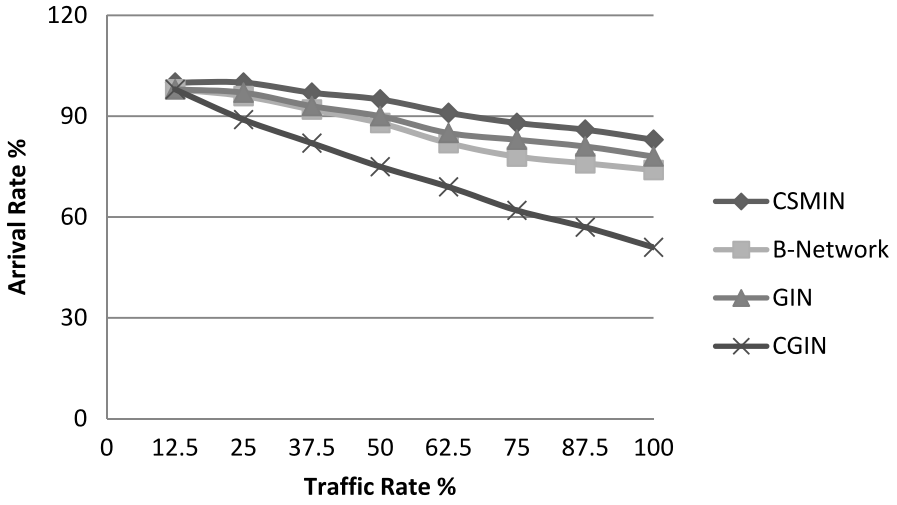
(a)

Fig. 5a The arrival rates vs. traffic load of CSMIN with backward straight links, B-Network, GIN, and CGIN for network size $N = 16$ and with the situation of no faults

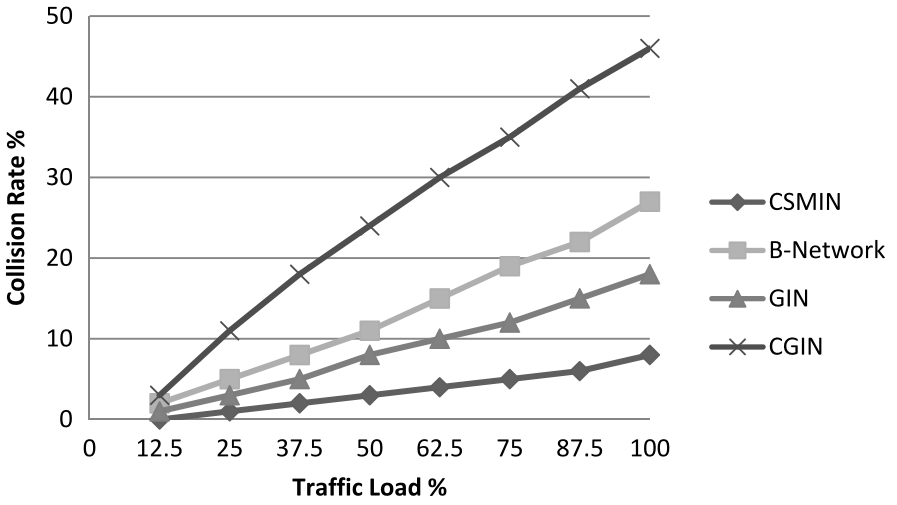
1. Network without faults;
2. In addition, whenever a faulty element exists in the network, we assumed the faulty switch was fully faulty (i.e. the faulty switch cannot receive or send any packets from any input links or output links.

For measuring the arrival rate (without switch fault), we preformed simulations on CSMIN with backward straight links, B-Network, GIN, and CGIN for $N = 16$ with various traffic loads (i.e. the number of packets that has been sent simultaneously by the different sources to different or same destination), which ranges between 12.5% to 100% to get our results and compared them. Furthermore, for measuring the arrival rate, collision rate and fault-tolerance rate (with switch faults), we preformed simulations on FCSMIN, CSMIN with backward straight links, B-Network, GIN, and CGIN for $N = 16$ with various traffic loads to get results for comparison.

Figure 5a presents the arrival rates of the networks with the situations of without fault. CSMIN with backward links have the best arrival ratio in comparison to B-Network, CGIN, and GIN for network size 16 and CGIN is the worst performer. On other hand, Fig. 5b–d presents the arrival rates, collision rates, and fault-tolerant rates of the networks with the situation of a fully faulty switch. In low traffic, GIN and CSMIN with backward straight links, which send two identical packets via two disjoint paths concurrently perform better arrival ratios than or almost equal arrival ratios by CGIN and B-Network as they can also tolerate a faulty switch. However, there is always rapid performance degradation because of packet collisions. The number of packet losses is very high for CGIN and very low for CSMIN. Therefore, CGIN present a worse arrival ratio than other networks when the traffic load is high. In addition, Fig. 5d presents the fault-tolerance capability of these networks. CSMIN with backward straight links have higher fault-tolerant ratios than GIN, CGIN, and B-



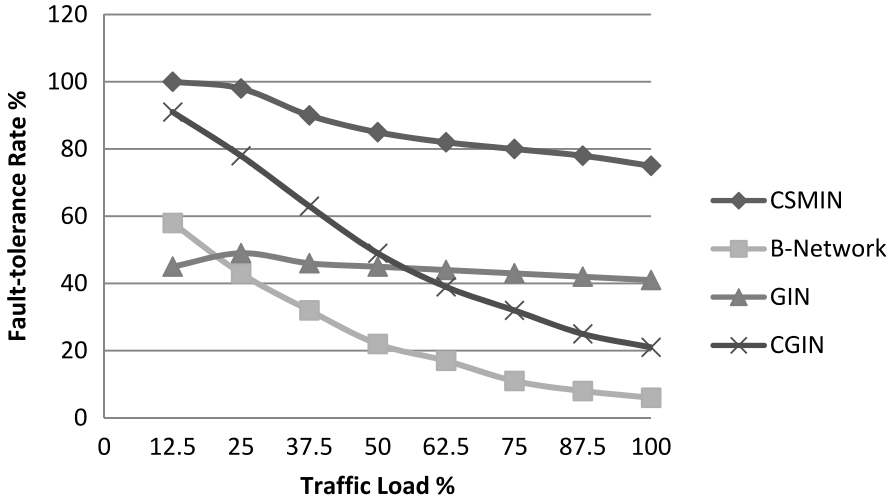
(b)



(c)

Fig. 5b–d The arrival rates, collision rates and fault-tolerance rates vs. traffic load of CSMIN with backward straight links, B-Network, GIN, and CGIN for network sizes $N = 16$ and with the situation of a fully faulty switch

Network. However, GIN performs better than CGIN and B-Network at higher loads (i.e. traffic load between 75% to 100%) but the collision losses are greater than the benefits from tolerating faults. However, CSMIN with backward straight links can tolerate faults and prevent collisions; moreover, it has a better arrival ratio than other networks.



(d)

Fig. 5b–d (*Continued*)

5 Fault-tolerant Fully-Chained Combining Switches Multi-stage Interconnection Network (FSCMIN)

To eliminate the backtracking penalties of CSMIN, we propose a new design called Fault-tolerant Fully-Chained Combining Switches Multi-stage Interconnection Network (FCSMIN). FCSMIN has the similar characteristics of 1-fault tolerance and two disjoint paths between any source-destination pair, but it can tolerate at least one link or switch fault at each stage without backtracking.

5.1 Topology of FCSMIN

FCSMIN has multiple paths between any source-destination pair to provide better fault-tolerance capability. The FCSMIN changes one of the original non-straight links of CSMIN at stage $i = 1$ to $n - 1$ to a chained link.

A FCSMIN of size $N = 2^n$ consists of $n + 1$ stages labeled from 0 to n. The first stage of FCSMIN is similar to CSMIN having 2×4 crossbar switches. For stages 1 to $n - 1$, each switch of FCSMIN is augmented with a chaining links. 3×3 switches replace the switches at intermediate stages. We also remove either of the non-straight links between the last two stages so that the final stage has 2×1 switches.

We have introduced this chaining links at all stages except for the last stage. Either the upward non-straight link or the downward non-straight link of CSMIN can be changed to a chaining link for stages 0 to $n - 1$. At the last stage n, we can remove either the upward non-straight link or the downward non-straight link. Thus, we present two models of FCSMIN namely, UpRoute-function based FCSMIN and DownRoute-function based FCSMIN. The UpRoute and DownRoute are two functions of the destination-tag routing algorithm, which is used by FCSMIN in order

to resolve the algorithmic and design issues faced by CSMIN. These functions are discussed later.

In FCSMIN, since the destination-tag routing algorithm does not involve backtracking, it uses bi-directional switches between stages 1 to n, and thus brought down the hardware cost of FCSMIN less than CSMIN.

In UpRoute function based FCSMIN, the chaining scheme is that switch j is chained to switch $(j - 2^i) \bmod 2^{n-1}$, where i denotes stage number from 1 to $n-1$ and $n = \log_2 N$. For example, at stage 1, the chain-out link of switch 2 is connected to the chain-in link of switch 0. In the last stage, we remove all the downward non-straight links. Figure 6 shows the topology of UpRoute function based FCSMIN.

In DownRoute function based FCSMIN, the chaining scheme is that switch j is chained to switch $(j + 2^i) \bmod 2^{n-1}$, where i denotes stage number from 0 to $n-1$ and $n = \log_2 N$. For example, at stage 1, the chain-out link of switch 2 is connected to the chain-in link of switch 4. In the last stage, we remove all the upward non-straight links. Figure 7 shows the topology of DownRoute function based FCSMIN.

5.2 Destination-tag routing

We have used two destination-tag routing functions UpRoute and DownRoute for routing packets from stage 1 to n in a FCSMIN. A switch j at stage i is an even switch if $j_i = 0$ or an odd switch if $j_i = 1$, where $j_0 j_1 \ldots j_{n-1}$ is the n-bit binary representation of j, and j_{n-1} is the most significant. Let T denote a destination-tag where $t_0 t_1 t_2 \ldots t_{n-1}$ is the binary representation of t and t_{n-1} is the most significant. By only using t_i, we can decide routing from the switch at stage i to the switch at stage $i+1$. The DownRoute function goes straight or downward, while the UpRoute function goes upward or straight. The behavior of UpRoute and DownRoute functions is depicted in Fig. 8 and represented by (1) and (2).

$$\text{UpRoute}(j, t_i) = \begin{cases} j + 2^i & \text{if } (j_i = 0 \text{ and } t_i = 1) \\ & \text{or } (j_i = 1 \text{ and } t_i = 0), \\ j & \text{otherwise} \end{cases} \tag{1}$$

$$\text{DownRoute}(j, t_i) = \begin{cases} j - 2^i & \text{if } (j_i = 0 \text{ and } t_i = 1) \\ & \text{or } (j_i = 1 \text{ and } t_i = 0). \\ j & \text{otherwise} \end{cases} \tag{2}$$

5.3 Routing scheme in FCSMIN

To implement the design of FCSMIN, we propose the use of a combination of distance-tag routing and destination-tag routing algorithm. The distance-tag routing algorithm is used to transfer packets between stage 0 and stage 1, while the destination-tag routing algorithm is used to transfer packets from stage 1 to stage n. Under this combinational scheme, the distance-tag algorithm generates a 2-bit routing-tag for transferring packets between stage 0 and stage 1. The four links to stage 1 from a 2×4 switch at stage 0 are of types downward non-straight, downward straight, upward straight, and upward non-straight. A 2-bit combination such as 00, 01, 10, and 11, respectively, represent these links.

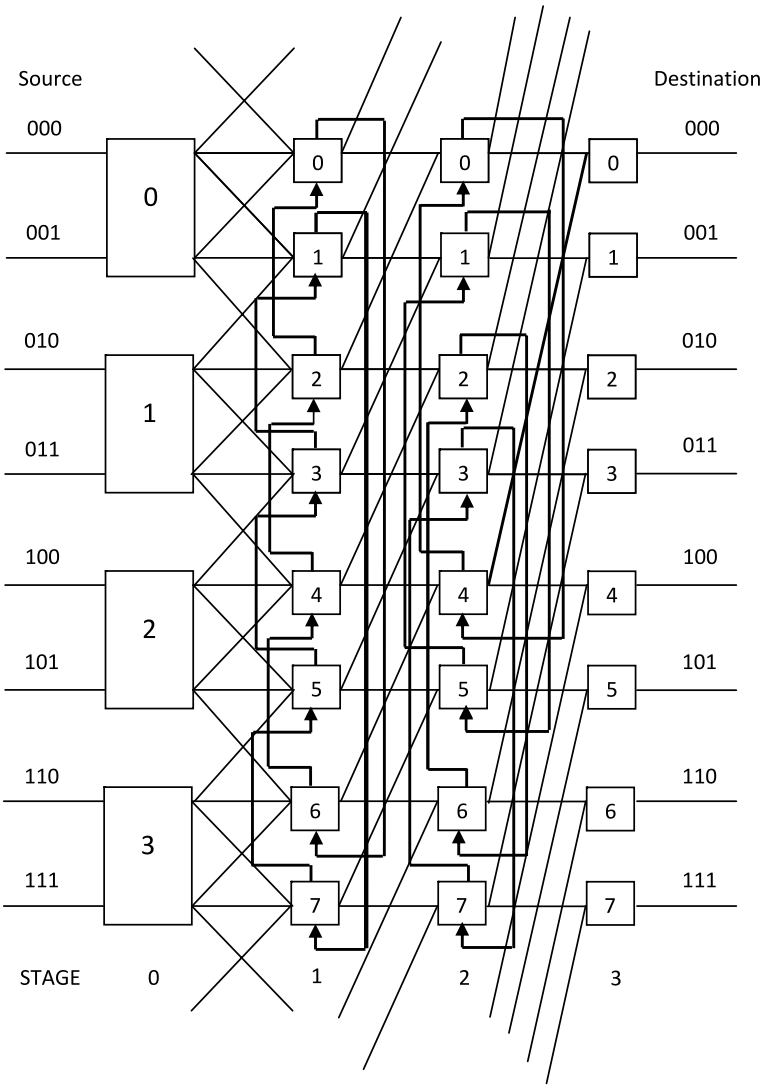


Fig. 6 UpRoute function based FCSMIN of size 8

For stages 1 to n with chaining links, the routing functions can be derived from the pre-defined UpRoute and DownRoute destination-tag routing functions as:

$$\text{UpRoute}(j, t_i) = \begin{cases} (j-1) \bmod N \text{ at stage } i \\ \qquad \text{if } (j_i = 0 \text{ and } t_i = 0) \\ \qquad \text{if } (j_i = 1 \text{ and } t_i = 0) \\ (j-1) \bmod N \text{ at stage } i+1 \\ \qquad \text{if } (j_i = 0 \text{ and } t_i = 0) \\ \qquad \text{if } (j_i = 1 \text{ and } t_i = 0) \end{cases}, \tag{3}$$

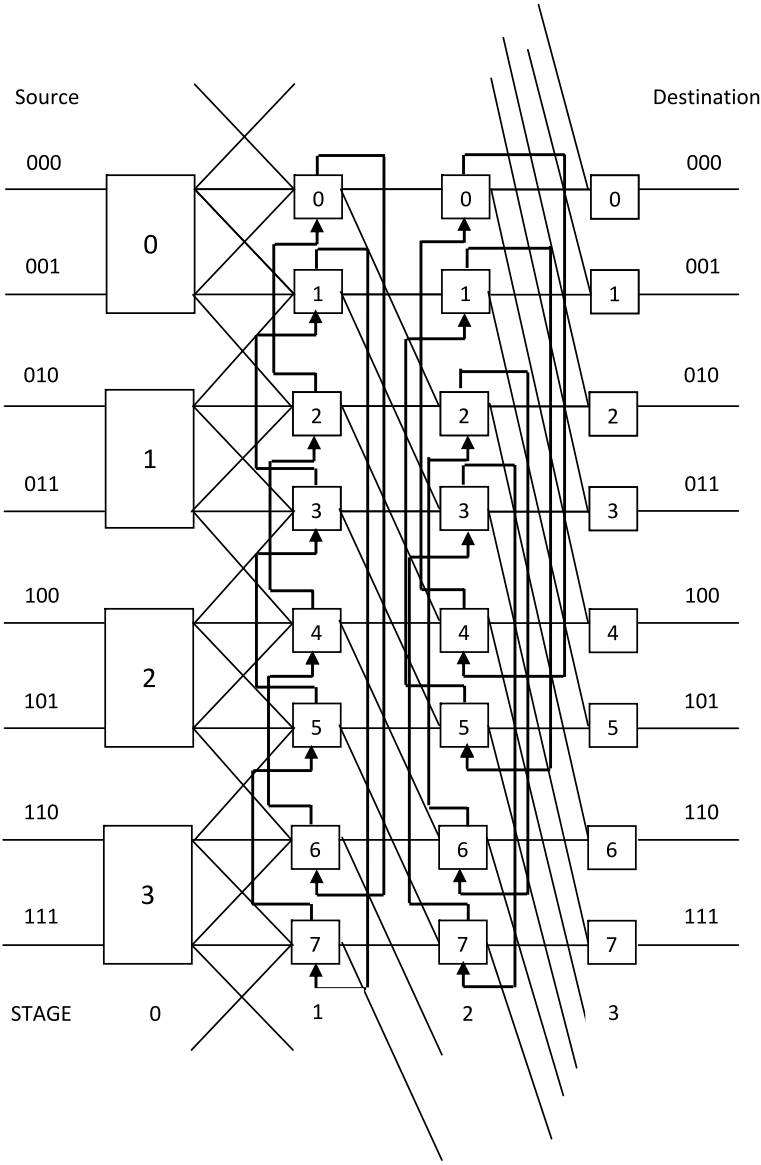


Fig. 7 DownRoute function based FCSMIN of size 8

$$\text{DownRoute}(j, t_i) = \begin{cases} (j+1) \bmod N \text{ at stage } i \\ \qquad \text{if } (j_i = 0 \text{ and } t_i = 0) \\ \qquad \text{if } (j_i = 1 \text{ and } t_i = 0) \\ (j+1) \bmod N \text{ at stage } i+1. \\ \qquad \text{if } (j_i = 0 \text{ and } t_i = 1) \\ \qquad \text{if } (j_i = 1 \text{ and } t_i = 1) \end{cases} \tag{4}$$

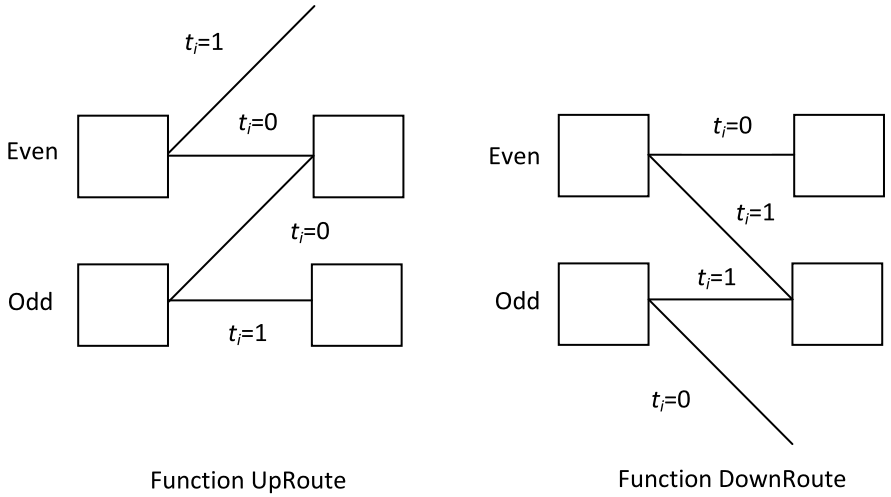


Fig. 8 Switching by UpRoute and DownRoute function at stage i

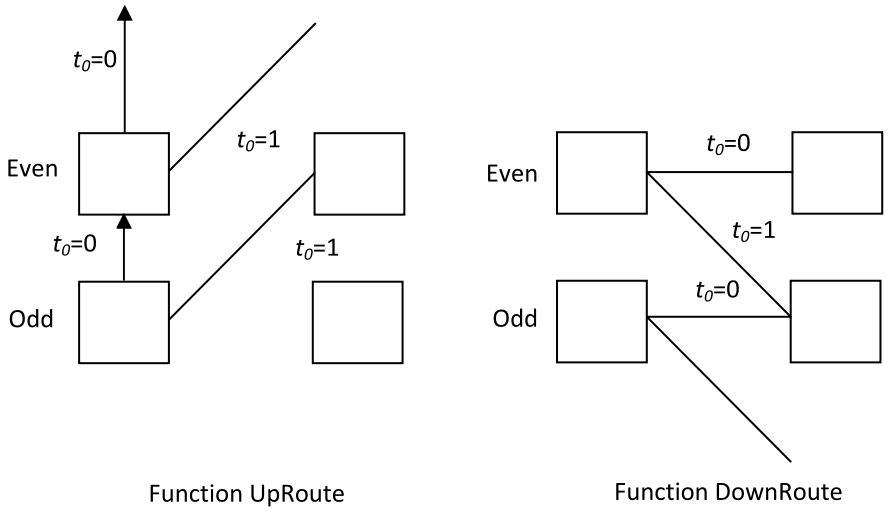


Fig. 9 Switching by UpRoute and DownRoute function for stages 1 to n

Let destination node be $D = d_0 d_1 d_2 \ldots d_{n-2} d_{n-1}$. Using the UpRoute function, when $d_i = 0$ in an odd switch at stage i, the packet is routed to an uplink else, the chaining link will be used as shown in Fig. 9. Using the DownRoute function, when $d_i = 1$ is in an odd switch at stage i, the packet is routed to a downlink else, the straight link will be used. In addition, when the switch is even, routing is opposite for both of the cases of UpRoute and DownRoute function applications.

When fault occurs, the chaining link provides alternate paths. Suppose a packet at a switch $j = j_0 j_1 j_2 \ldots j_{n-1}$ encounters a fault in the link between stage i and stage $i+1$, the packet should be routed via the chaining link at stage i to switch $j - 2^i \bmod 2^n$, for $1 \leq i \leq n-1$ where $n = \log_2 N$. Thus, the FCSMIN has a strong capability to tolerate the link fault at each stage and allows dynamic link rerouting.

5.4 New approach of providing fault tolerance in CSMIN using multiplexers and demultiplexers

CSMIN is a 1-fault tolerant MIN for all source-destination pairs; it guarantees fault-tolerance only for the intermediate stages. There are no alternative paths in the network in case of faults at stage 0 and stage n. Hence, in case of failure of switches or links at the first or last stages in CSMIN, the packets will be lost completely. In order to make the network 1-fault tolerant at all stages (increasing the degree of fault-tolerance in CSMIN), we had proposed new modifications in the existing design of FCSMIN by removing the chaining links and by introducing multiplexers and demultiplexers. Figure 10 shows the design of the new CSMIN of size 8 and its topology is described as follows:

1. Before stage 0, 2:1 multiplexers are introduced for each source i.
2. After last stage, 1:2 demultiplexers are introduced for each destination i.

5.4.1 Routing and rerouting in CSMIN with multiplexers and demultiplexers (CSMINMD)

The function of adding multiplexers and demultiplexers at first and last stage of CSMIN are to facilitate fault-tolerance and at those stages. For 1-fault tolerance at the intermediate stages, the rerouting algorithm of CSMINMD is as follows.

The topology of the network is such that in case of a switch or link failure at stage 0, the corresponding 2:1 multiplexer will transfer the packet on another input line and then is routed to its destination from the new input line. In case of a switch or link failure at stage n, the rerouting algorithm of CSMINMD transfers the packet to the pre-stage switch of the alternative path. The packet is then transferred from this switch by the straight link to a switch at stage n whose corresponding 1:2 demultiplexers has the capability to transfer the packet to the original destination. By exploiting the topology of CSMIN, the selection of new input and output lines is in such a way that the packet always reaches its original destination. Example 4 helps in illustrating this fault-tolerance capability at stage 0 and stage n.

Example 4 The source is 6, the destination is 4, and suppose that the switch 3 at stage 0 and the switch 4 at stage 3 is faulty. The routing and rerouting paths are described as follows.

Solution: In that case, the multiplexer connected to input line 6 would transfer the packet to its alternative input line 2 and then is routed for destination 4. Suppose the downward path is followed to reach destination 4 from new source 2. On encountering

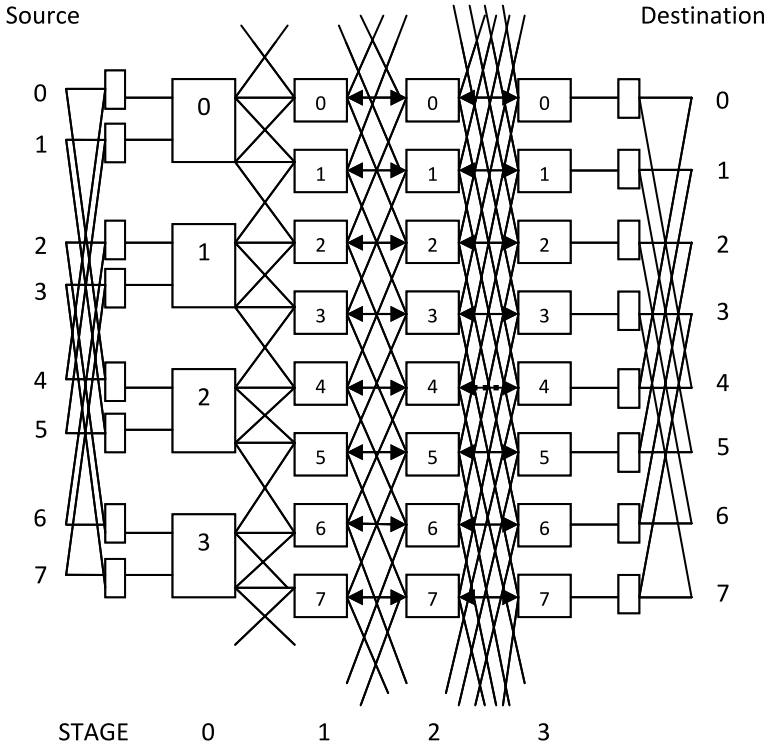


Fig. 10 Combining switches interconnection network with multiplexers and demultiplexers (CSMINMD)

the faulty switch 4 at stage 3, then the packet is transferred to the pre-stage switch 0 of the upward path by the rerouting algorithm of CSMINMD. The switch 0 at stage 2 then transfers the packet to switch 0 at stage 3 by the straight link. The demultiplexers connected to the switch 0 at stage 3 transfers the packet to its alternative output line 4. Hence, the packet is not lost and arrives at the original destination 4. Figure 11 illustrates this rerouting behavior for Example 4.

In the next section, we simulated and compared the FCSMIN, CSMIN, B-Network, GIN, and CGIN under without faults, with a switch fault, and with two switch faults to get the arrival ratio and fault-tolerant ratio.

6 Experimental setup and comparative analysis of FCSMIN, CSMIN, B-Network, GIN, and CGIN based on simulation results

The issues related to CSMIN, which resulted in low fault-tolerance, high hardware costs, and increased system latency have been addressed here.

For our simulation, we have used an IBM System x, running with Novell's SUSE Linux Enterprise Server 11, and we continuously and randomly generated the different source-destination requests in each cycle and continuously ran 32,768 cycles

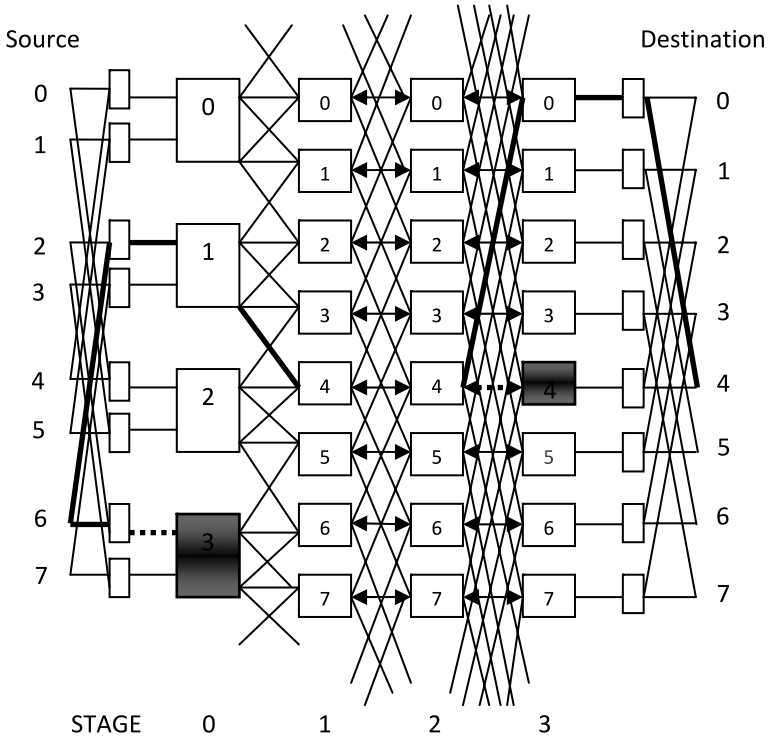


Fig. 11 Routing and rerouting in CSMINMD for source 6 and destination 4 as shown by the *bold lines*. The faulty switch is indicated by the *gray color* and the *dashed line* means the original routing path

to compute the arrival rate and fault-tolerance rate. The following cases have been addressed while computing the fault-tolerance rate:

1. Network without faults;
2. In addition, whenever a faulty element exists in the network, we assumed the faulty switch was fully faulty.

For measuring the arrival rate (without switch fault), we preformed simulations on FCSMIN, CSMIN with backward straight links, B-Network, CGIN, and GIN for $N = 16$ with various traffic loads, which ranges between 12.5% to 100% to get our results and compared them. In addition, for measuring the fault-tolerance rate (with switch faults), we preformed simulations on FCSMIN, CSMIN with backward straight links, B-Network, CGIN, and GIN for network sizes $N = 16, 32$ and 64 (with various traffic loads) to get results for comparison.

6.1 Fault-tolerance

Figure 12a–c, shows arrival rates (without switch fault, with a switch fault, and with two switch faults) vs. traffic load of FCSMIN, B-Network, CGIN, CSMIN with backward straight links, and GIN when the network size is 16. From the graph, it is depicted that the FCSMIN has a better arrival ratio in comparison to other MINs. The

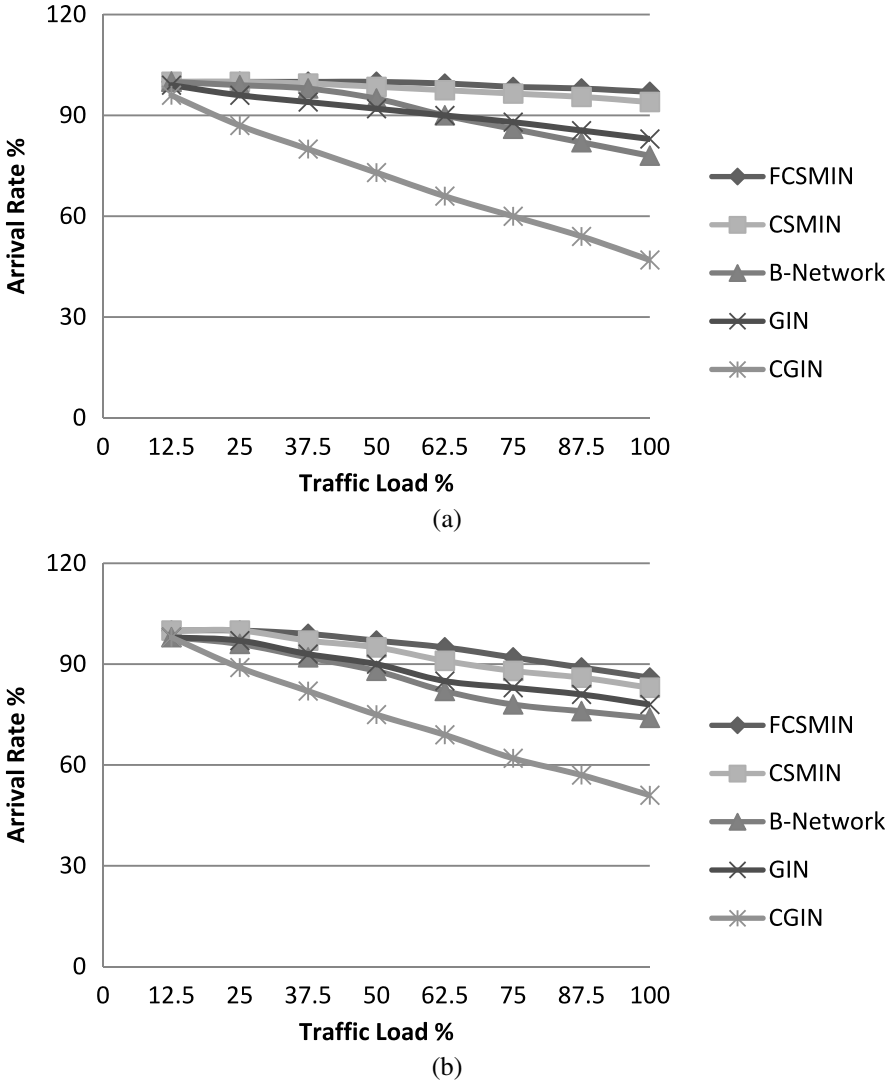
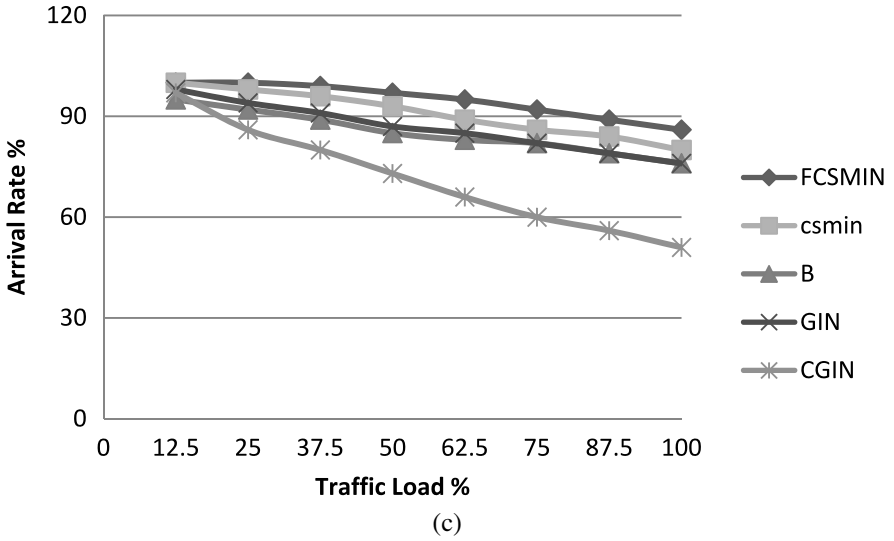


Fig. 12a–c Arrival rate (without switch fault), arrival rate (with a switch fault), and arrival rate (with two switch faults) of FCSMIN, CSMIN with backward straight links, B-Network, GIN, and CGIN for network size $N = 16$

performance B-Network and GIN in terms of arrival ratio is comparable and CGIN always has a poor arrival ratio for all three cases.

On the other hand, Fig. 12d–f presents the fault-tolerance capability of (with a switch fault) of FCSMIN, CSMIN with backward straight links, B-Network, GIN, and CGIN when the network size is 16, 32, and 64. As the size increases, FCSMIN provides better fault-tolerance capability in comparison to every network especially from CSMIN. Moreover, CSMIN is not better than FCSMIN but it tolerates a greater number of faults than B-Network, GIN, and CGIN. At the traffic load, which ranges



(c)

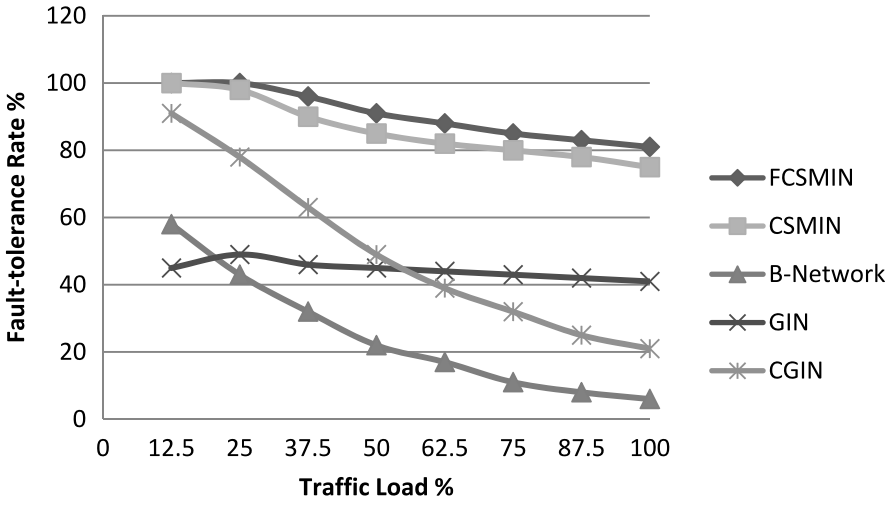
Fig. 12a–c (*Continued*)

between 62.5% to 100%, GIN is far better than B-Network. CGIN network comes out to be the worst performer in terms of fault-tolerance. However, the collision losses are greater than the benefits from tolerating faults. From here, we can conclude that FCSMIN can tolerate faults and prevent collisions; moreover, it has a better arrival ratio than other networks.

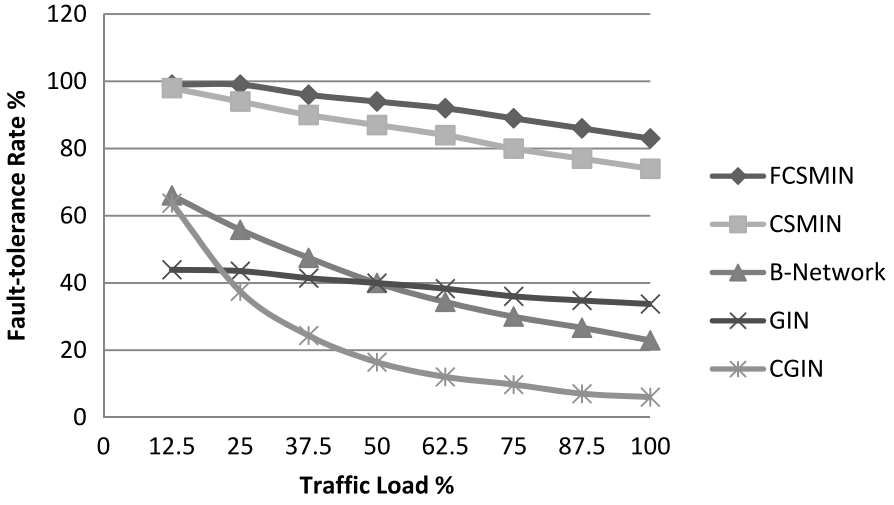
Furthermore, Fig. 12g–i presents the fault-tolerance capability of (with a two-switch fault) of FCSMIN, CSMIN with backward straight links, B-Network, GIN, and CGIN when the network size is 16, 32, and 64, separately. For all sizes, FCSMIN remains better in comparison to CSMIN and best when compared to others. GIN remains the outperformer when compared with B-Network and CGIN remains a poor performer and eventually cannot tolerate too many faults.

6.2 Hardware cost

The switch hardware complexity of CSMIN is high because bi-directional switches have been used between stages 1 to n in order to backtrack a packet to the previously traversed switch $(i - 1)$ in case of a fault or collision at stage i. The use of these switches (to solve the backtracking) increases the hardware cost of the network. FCSMIN does not make use of backtracking mechanism, therefore, use of bi-directional switches is eliminated. In addition to this, one of the non-straight links is removed from the switches in stages 1 to n depending upon the functionality of FCSMIN being followed by using either UpRoute or DownRoute functions. This results in further cost reduction of FCSMIN. If we compute the cost of CSMIN and FCSMIN, taking the cost of k, $m \times n$ switches as $k \times m \times n$ units, the cost of uni-directional links as 1 unit and the cost of bi-directional links as 2 units, then we observe that CSMIN costs 376 units whereas FCSMIN costs 264 units for networks of size $N = 8$. From Fig. 13, it is depicted that as the size of the network increases the hardware cost of the CSMIN



(d)



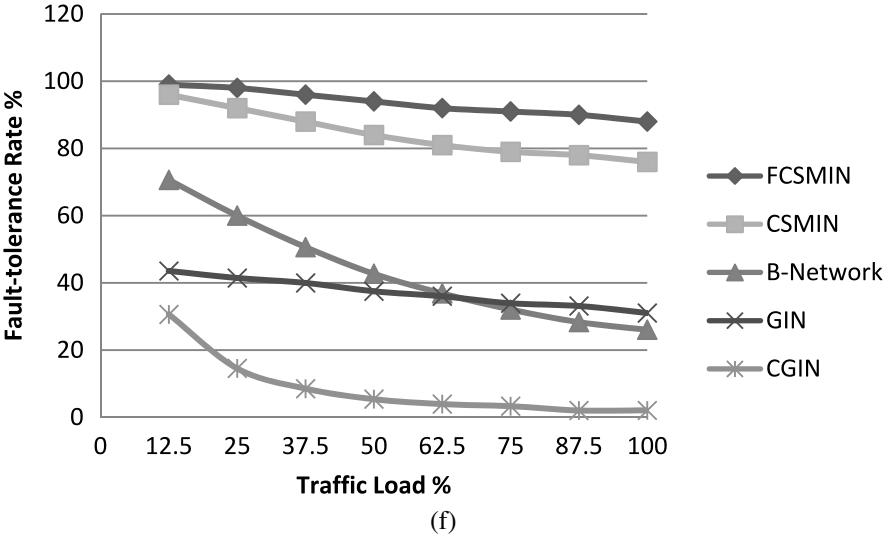
(e)

Fig. 12d–f Fault-tolerance rates (with a switch fault) vs. traffic load of FCSMIN, CSMIN with backward straight links, B-Network, GIN, and CGIN for network sizes $N = 16$, 32, and 64 separately

becomes higher in comparison to the hardware cost of FCSMIN. On the lower level, the hardware cost of CSMIN is comparable to FCSMIN but for the higher network sizes. There is a significant increase in the cost of CSMIN. Thus, our proposed design of FCSMIN is cheaper to implement than CSMIN.

6.3 System latency

The distance-tag algorithm used in CSMIN involves pre-processing overhead to compute the $n + 1$ bit routing-tag. Our proposed algorithm uses distance-tag routing only



(f)

Fig. 12d–f (*Continued*)

between stage 0 and 1. This approach requires computation of a 2-bit routing-tag before the packet transfer takes place. Thus, our proposed algorithm results in a less amount of pre-processing overhead. Moreover, the use of destination-tag routing frees our routing scheme from the need to use backtracking to deal with faults or collisions. This results in a significant improvement in system latency. Thus, FCSMIN possesses the same 1-fault tolerant and two disjoint path features of CSMIN at a lower hardware cost and achieves routing and rerouting with better performance metrics.

7 Conclusion

The 1-fault tolerant CSMIN provides two disjoint paths to solve collision situation. In this paper, we have presented more efficient and comprehensive algorithms that provide two disjoint paths between every source-destination pair that reach the correct destination. Since the vertical distance between these two paths at any particular stage i is 2^i, the switch can easily reroute the packet between these two stages, in the event of a fault or collision. However, the proposed algorithm backtracks packet in case of a faulty switch or collision, and eventually it reduces network operational cost. Moreover, to achieve a good arrival ratio with respect to increasing traffic load, it is imperative to keep the packets loss due to collisions and faults, as low as possible. The need to lower the hardware costs and backtracking penalties drove us to make the necessary changes in CSMIN. To deal with such cases, we have modified the existing design of CSMIN and presented an alternative design FCSMIN. Hardwiring of chaining links in FCSMIN for stages 1 to $n-1$ eliminates the need of other redundant hardware used in CMSIN. It also retains the two disjoint paths that successfully exhibit by the topology of CSMIN. The use of destination-tag algorithm helps us to eliminate backtracking in case of faults or collisions. Furthermore,

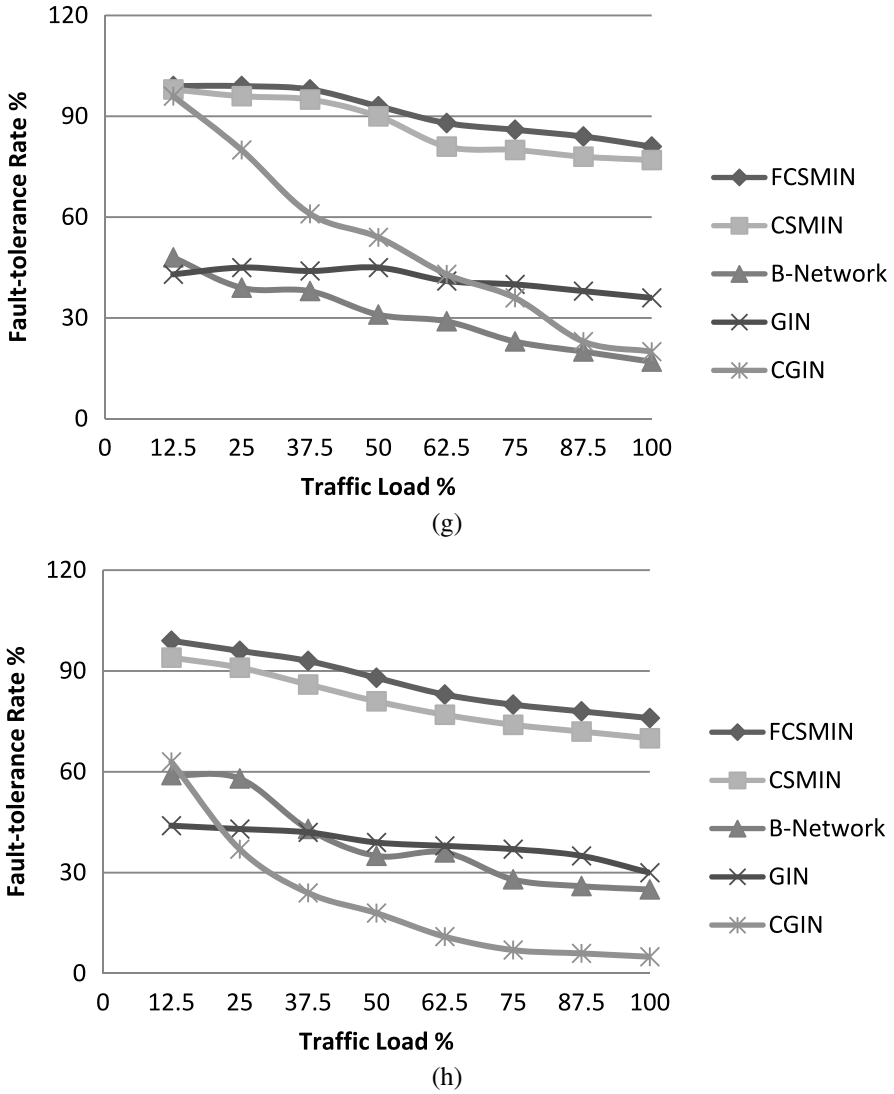
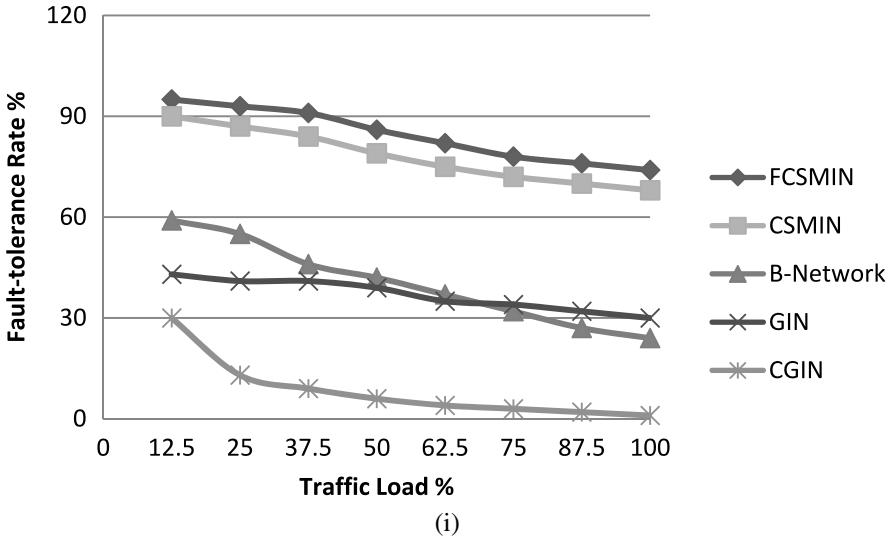


Fig. 12g–i Fault-tolerance rates (with two switch faults) vs. traffic load of FCSMIN, CSMIN with backward straight links, B-Network, GIN, and CGIN for network sizes $N = 16$, 32, and 64 separately

FCSMIN fault-tolerance ratio with one and two switch faults is higher in comparison to CSMIN, B-Network, GIN, and CGIN. Therefore, FCSMIN is 1-fault tolerant MIN with a lower hardware cost and provides better system latency than CSMIN and other disjoint path MINs.



(i)

Fig. 12g–i (*Continued*)

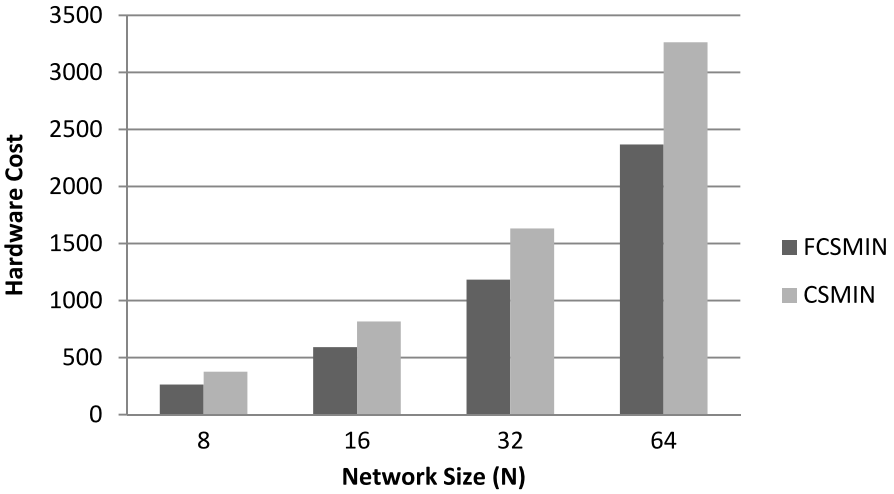


Fig. 13 Comparison of FCSMIN and CSMIN with backward straight links based on hardware cost

8 Future scope

Our proposed algorithm for routing packets on CSMIN and FCMSIN achieves 1-fault-tolerance to solve the faults or collision situations. To achieve a good arrival ratio in terms of increasing traffic, it is imperative to keep the packet loss (due to collisions and faults), as low as possible. For this, we will be attempting to design a comprehensive and efficient algorithm that will make the FCSMIN 2-fault tolerant.

Moreover, we are working on to provide three disjoint paths in the FCMIN by making necessary hardware changes to its existing design.

Acknowledgements Very special thanks to Jaypee Education System (JES) and Jaypee University of Information Technology (JUIT), for providing excellent Research Environment and Learning Ambience and Sun Solaris and High-end Parallel Computing Lab at JUIT.

This research work is dedicated to my Shri Shri 1008 Swami Shri Paramhans Dayal Sacchidanand Maharaja Ji and the loving memories of my departed maternal grandfather and grandmother and father-in-law, who continue to guide me in spirit.

References

1. Feng TY (1981) A survey of interconnection networks. IEEE Comput 14:12–27
2. Hwang K, Briggs FA (1984) Computer architecture and parallel processing. McGraw-Hill, New York, ISBN 0-07-066354-8
3. Adams GB III, Agrawal DP, Siegel HJ (1987) A survey and comparison of fault-tolerant multi-stage interconnection networks. IEEE Comput 20:14–27
4. (1987) Special issue on interconnection networks, IEEE Comput 20
5. Siegel HJ (1990) Interconnection network for large scale parallel processing: theory and case studies. McGraw-Hill, New York, ISBN 0-07-057561-4
6. Hwang K (2000) Advanced computer architecture: parallelism, scalability, programmability. Tata McGraw-Hill, Delhi, ISBN 0-07-053070-X
7. Duato J, Yalamanchili S, Ni LM (2003) Interconnection networks: an engineering approach. Morgan Kaufmann, San Mateo, ISBN 1-55860-852-4
8. Dally W, Towles B (2004) Principles and practices of interconnection networks. Morgan Kaufmann, San Francisco, ISBN 978-0-12-200751-4
9. Nitin, Subramanian A (2008) Efficient algorithms to solve dynamic MINs stability problems using stable matching with complete TIES. J Discrete Algorithms 6(3):353–380
10. Nitin, Sehgal VK, Sharma N, Krishna K Bhatia A (2007) Path-length and routing-tag algorithm for hybrid irregular multi-stage interconnection networks. IEEE Computer Society Press, Los Alamitos, pp 652–657
11. Nitin, Sehgal VK, Bansal PK (2007) On MTTF analysis of a fault-tolerant hybrid MINs. WSEAS Trans Comput Res 2:130–138
12. Nitin (2006) Component level reliability analysis of fault-tolerant hybrid MINs. WSEAS Trans Comput 5:1851–1859
13. Nitin (2006) Reliability analysis of multi-path multi-stage interconnection network. In: Proceedings of the 10th WSEAS international conference on circuits, systems, communication and computers, 2006, pp 1018–1023
14. Nitin (2006) On analytic bounds of regular and irregular fault-tolerant multi-stage interconnection networks. In: Proceedings of the international conference on parallel and distributed processing techniques and applications, 2006, pp 221–226
15. Nitin, Subramanian A (2006) On reliability analysis of cost-effective hybrid zeta network: a fault-tolerant multi-stage interconnection network. In: Proceedings of the international conference on parallel and distributed processing techniques and applications, 2006, pp 260–265
16. Subramanian A, Nitin (2004) On a performance of multi-stage interconnection network. In: Proceedings of the 12th international conference on advanced computing and communication, 2004, pp 73–79
17. Nitin, Chauhan DS, Sehgal VK (2008) Two $O(n^2)$ time fault-tolerant parallel algorithm for inter NoC communication in NiP. Springer, Berlin, pp 262–287
18. Sehgal VK, Nitin (2007) Stochastic communication on application specific networks on chip. Springer, Berlin, pp 11–16, ISBN 978-1-4020-6265-0
19. Chen CW, Chung CP (2005) Designing a disjoint path interconnection network with collision solving and fault tolerance. J Supercomput 34(1):63–80
20. Chen CW (2006) Design schemes of dynamic rerouting networks with destination tag routing for tolerating faults and preventing collisions. J Supercomput 38(3):307–326

21. Siegal HJ, Jose DR, Fortes AB (1992) Destination tag routing techniques based on a state model for the IADM network. IEEE Trans Comput 41(3):274–285
22. Smith B (2006) Design of dynamic rerouting networks with destination tag routing for tolerating faults and preventing collisions. Springer, Berlin
23. Parker DS, Raghavendra CS (1984) The gamma network. IEEE Trans Comput 33:367–373
24. Chuang PJ (1996) CGIN: a fault tolerant modified gamma interconnection network. IEEE Trans Parallel Distrib Syst 7(12):1301–1306
25. Chen CW, Lu NP, Chen TF, Chung CP (2000) Fault-tolerant gamma interconnection networks by chaining. IEE Proc Comput Digital Tech 147(2):75–80
26. Seo SW, Feng TY (1995) The composite banyan network. IEEE Trans Parallel Distrib Syst 6(10):1043–1054
27. Lee KY, Yoon H (1990) The B-Network: a multistage interconnection network with backward links. IEEE Trans Comput 39(7):966–969
28. Lau FCM, Poon WC (1998) Throughput analysis of B-Networks. IEEE Trans Comput 47(47):482–485