# Wormhole Routing on the Star Graph Interconnection Network *

Chang-Biau Yang and Tung-Hsing Liu

Department of Applied Mathematics,
National Sun Yat-sen University
Kaohsiung, Taiwan 804, Republic of China
e-mail: cbyang@math.nsysu.edu.tw

**Abstract.** A deadlock-free wormhole routing algorithm can be developed by using the concept of virtual channels. In this paper, we first propose two wormhole routing algorithms on the star graph. The first one is minimal, fully adaptive, and deadlock-free and it requires $\lfloor \frac{3n+1}{4} \rfloor$ virtual channels for per physical link on an $n$-star graph. For the second algorithm, we make some restrictions on the routing path selection. Thus, it reduces the number of virtual channels to $\lfloor \frac{n+1}{2} \rfloor$. It is still minimal and deadlock-free. However, it is only partially adaptive. Both of these results have significant improvement on the previous results. Then we propose a fault-tolerant wormhole routing algorithm, which can tolerate one fault in the star graph. The length of the routing path is at most $d+2$, where $d$ is the distance between the source and the destination. And it does not need any extra virtual channel.

## 1 Introduction

The performance of an interconnection network depends on switching techniques and routing algorithms. There are two major and closely related classifications of a routing algorithm, one is minimal or non-minimal, and the other is non-adaptive, partially adaptive, or fully adaptive. A routing method is *minimal* if it always transmits a message via one shortest path from its source to its destination; while a *non-minimal* one allows to use longer paths to transmit a message. *Non-adaptive* routing is also called the deterministic routing, and the routing path is predetermined by the source and the destination. A routing algorithm is *adaptive* if alternative paths between a pair of source and destination are allowed to travel. *Fully adaptive minimal routing* may use any one of shortest paths to transmit a message between the source and the destination. And it is *partially adaptive minimal routing* if some of shortest paths may not be used.

In *wormhole routing* [13], a message is decomposed into a sequence of flits, where a *flit* is the smallest unit of information that a queue or channel can accept. Flits are routed in a pipelined fashion. Only the first flit of a message contains

* This research work was partially supported by the National Science Council of the Republic of China under contract NSC 87-2213-E-110-004.

all the routing information needed to decide the next channel to forward, and the rest flits follow the previous flit to advance. If the next channel is available, the header flit moves to that channel; otherwise, all of the flits block, and remain in the current channel. In realizing wormhole routing, each channel has a small buffer which is used to store a flit. Compared to store-and-forward, wormhole routing reduces the latency of message delivery if the traffic on the network is light. Another advantage is that, in wormhole routing, a node does not need to allocate the buffer for an entire packet. Thus, it reduces the amount of buffer required on each node. Then, it makes possible to build small, fast and cheap routers.

One of the most important issues in the routing algorithm is deadlock avoidance. Wormhole routing causes deadlock more easily, since in wormhole routing, a blocked message does not relinquish the channels it has already occupied. The deadlock problem can be solved by using the concept of *virtual channels* [4]. Virtual channels are logical channels which use the time-division multiplexing method to share a physical channel. Each virtual channel of each physical link to each node has its own queue. However, the more virtual channels are used, the larger buffers and bandwidth associated with each physical channel are needed, and then the cost for communication is increased.

An $n$-dimensional star graph, shortened as $n$-star and denoted by $S_n$ [1], is an edge- and node-symmetric graph containing $n!$ nodes and $\frac{(n-1)n!}{2}$ edges. The nodes are labeled by the $n!$ permutations on $n$ symbols $(1, 2, \cdots, n)$. For any pair of nodes, there is an edge between them if and only if their labels differ only in the first(leftmost) position and one other position. For example, in a 4-star, which contains $4! = 24$ nodes, nodes 1234 and 2134 are connected by a link. Similarly, 1234 and 3214 are also neighbors. Figure 1 shows a 4-star. $g_i$ is defined as $g_i(p_1p_2\cdots p_n) = (p_ip_2p_3\cdots p_{i-1}p_1p_{i+1}\cdots p_n)$, where $p_1p_2\cdots p_n$ is a permutation of $123\cdots n$ and $2 \leq i \leq n$. If an edge is labeled with $g_i$, then it connects two nodes $A$ and $B$, where $g_i(A) = B$ and $g_i(B) = A$. For instance, in $S_4$, $g_2(1234) = 2134$, $g_3(1234) = 3214$, and $g_4(1234) = 4231$.

Recently, many wormhole routing algorithms have been proposed [3, 5, 8]. And some of them are fault tolerant [9]. However, most of them are designed for the $k$-ary $n$-cube based on the torus and meshes [5, 7, 10]. Only few wormhole routing algorithms have developed on the star graph [12, 14].

Misic and Jovanovic proposed a minimal, adaptive and deadlock-free wormhole routing algorithm in $S_n$ [12]. Their approach divides a physical channel into $n-1$ virtual channels. Virtual channels sharing the same physical channel are numbered and the virtual channels are allocated according to a decreasing order.

Ravikumar and Goel proposed two deadlock-free wormhole routing algorithms for $S_n$ [14]. They also use the concept of virtual channels to avoid deadlock. In the first algorithm, for each physical channel, it needs at most $n-1$ and at least one virtual channel. And the average number of virtual channels per physical channel is $\frac{n}{2}$. However, the routing path is non-minimal. In the second algorithm, the routing path is minimal, the maximal number of virtual channels per physical channel is $\lfloor\frac{3(n-1)}{2}\rfloor$, and the average number of virtual channels per

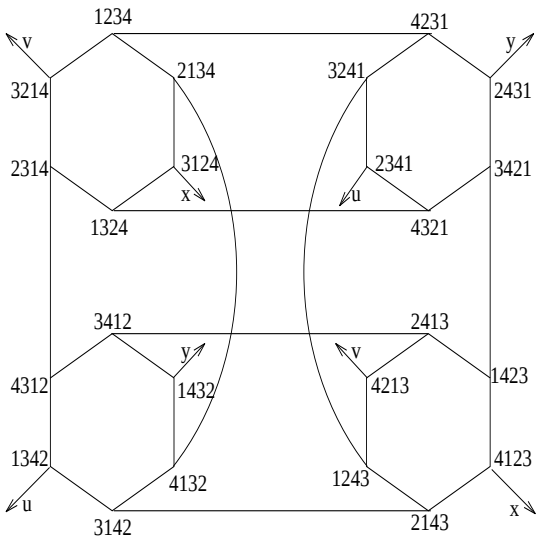


**Fig. 1.** A 4-dimensional star graph, $S_4$.

physical channel is $\lfloor \frac{5n}{4} \rfloor - 1$.

In this paper, we shall first propose two wormhole routing algorithms on the $n$-star graph, both of which are minimal. The first one is fully adaptive and needs $\lfloor \frac{3n+1}{4} \rfloor$ virtual channels per physical channel. And, the second one is partially adaptive and needs $\lfloor \frac{n+1}{2} \rfloor$ virtual channels per physical channel. Compared with the previous results, both of our algorithms improve the number of virtual channels.

We also consider that there is at most one faulty node on the star graph. To make it be one-fault tolerant, the length of the routing path may be not minimal. Most previous fault-tolerant wormhole routing methods are based on the addition of virtual channels. We propose a method that the routing path is of length no more than $d+2$, where $d$ is the distance between the source and the destination. The above two of our wormhole routing algorithms can be applied under the situation that no extra virtual channel is required.

Due to the page limitation, some lemma and theorem proofs are omitted. The rest of this paper is organized as follows. In Section 2 we define the *symbol cycle structure*, instead of commonly used *cycle structure*, to represent the permutation between the source and destination nodes. Then, we show some operations on the symbol cycle structure. In Sections 3 and 4, we shall propose two wormhole routing algorithms, one is fully adaptive and the other is partially adaptive. And, in Section 5, we propose a wormhole routing algorithm which can tolerate one fault without increasing any virtual channel. And finally, a conclusion is given in Section 6.

## 2 The Symbol Cycle Structure

A permutation can be represented the cycle structure, which is formed by the product of several permutation cycles. For example, permutation 615342 can be represented as (621)(543), which means that symbol 4 occupies position 5, symbol 3 occupies position 4, and so on. Since each node in a star graph is represented by a permutation, the routing between two nodes is equivalent to sort the source permutation to the destination permutation. The result of sorting is obtained in the form of a product of generators $g_i's$. Since the star graph is node symmetric, the routing between two arbitrary nodes in the $S_n$ can be reduced to the routing from the corresponding node to the identity node $123\cdots n$.

Day and Tripathi proposed two rules to route from a permutation $P$ to the identity node in $S_n$ [6]. The two rules are given as follows.

(1) If the first(leftmost) symbol of permutation $P$ is 1, then exchange it with any symbol not in its correct position.

(2) If the first symbol of $P$ is $j \neq 1$, then either exchange it with the symbol at position $j$ or exchange it with any symbol in a cycle of the cycle structure of length at least two, other than the cycle containing $j$.

They also showed that these two rules ensure a routing path with minimum length from an arbitrary node to the identity node. The length of the above routing path can be calculated as follows.

$$d(P) = c + m - \begin{cases} 0 & : \ if\ 1\ is\ in\ the\ first\ position, \\ 2 & : \ if\ 1\ is\ not\ in\ the\ first\ position, \end{cases}$$

where $c$ denotes the number of cycles of length at least two and $m$ is the number of symbols in these cycles.

For example, in $S_6$, suppose 615342 is the source node, and the destination node is the identity 123456. Then the cycle structure is (6 2 1)(5 4 3), and one of the shortest routing paths is given as follows.
$(615342) \xrightarrow{g_6} (215346) \xrightarrow{g_2} (125346) \xrightarrow{g_3} (521346) \xrightarrow{g_5} (421356) \xrightarrow{g_4} (321456) \xrightarrow{g_3}$ (123456)

As another example, in $S_6$, suppose 643512 is the source node, and the destination node is 425136. Then the cycle structure of the permutation 643512 with respect to 425136 is also (6 2 1)(5 4 3). Thus, one of the shortest routing paths can be given by the same $g_i$ sequence as follows.
$(643512) \xrightarrow{g_6} (243516) \xrightarrow{g_2} (423516) \xrightarrow{g_3} (324516) \xrightarrow{g_5} (124536) \xrightarrow{g_4} (524136) \xrightarrow{g_3}$ (425136)

We use $T[j]$ to denote the $j$th symbol of a node $T$, where the leftmost symbol is counted as 1, i.e. $T[1]$ denotes the leftmost symbol. The *polarity* of one link, from node $A$ to node $B$, is defined to be positive if $A[1] < B[1]$; otherwise, it is negative. Note that the link polarity from $A$ to $B$ and the link polarity from $B$ to $A$ are complement to each other. In addition, $(i)*$ represents a node whose leftmost symbol is $i$.

To simplify the description of our algorithms, we define another relation between two nodes, called the *symbol cycle structure*, shortened as $SCS$ and

denoted by <>. In the cycle structure of a permutation, the symbols represent the relative positions between two permutation; while the symbols in an $SCS$ indicate the absolute positions of two permutations. For example, suppose that the source node and the destination node are 643512 and 425136 respectively. Then the cycle structure is (6 2 1)(5 4 3) and the corresponding $SCS$ is $< 6\ 2\ 4 >< 3\ 1\ 5 >$. It means that the symbols 6, 2 and 4 are on misplaced positions, and symbol 1, 5 and 3 are also on misplaced positions. In $< 3\ 1\ 5 >$, it means that symbol 1 occupies the position of symbol 3, symbol 5 occupies the position of symbol 1 and symbol 3 occupies the position of symbol 5. While (5 4 3) means that the symbol on position 5 occupies the position of the symbol on position 3, the symbol on position 4 occupies the position of the symbol on position 5, and so on. If we rotate symbols 6, 2 and 4, and 3, 1 and 5 respectively, we will get the routing path from the source to the destination.

Function $f_{p_i}$ is defined as $f_{p_i}(p_1\ p_2\ \cdots\ p_n) = (p_i\ p_2\ p_3\ \cdots\ p_{i-1}\ p_1\ p_{i+1}\ \cdots\ p_n)$, where $p_1\ p_2\ \cdots\ p_n$ is a permutation of $123\cdots n$ and $2 \leq i \leq n$. In other words, if $f_{p_i}$ is performed, symbol $p_i$ is exchanged with the leftmost symbol. Note that $f_j$ exchanges symbol $j$ with the leftmost symbol and $g_j$ exchanges the symbol on position $j$ with the leftmost symbol. A function $f_{p_i}$ is shortened as $f$ if there is no ambiguity. Since the leftmost symbol of the destination node may not be on its correct position, we attach a minus sign to the leftmost symbol of the destination node to denote this special symbol. For example, if the source node and the destination node are 643512 and 425136 respectively, the exact $SCS$ is $< 6\ 2\ -4 >< 3\ 1\ 5 >$.

Given an $SCS$, we will show how to route, via one shortest path, from the source node to the destination node by performing function $f$. Let the $SCS$ between $S$ and $D$ consists of $t$ symbol cycles $Y_1, Y_2, \cdots, Y_t$. And let $Y_1 =< a_1\ a_2\ \cdots\ a_r\ -x_1 >$, and $Y_i =< y_{i,1}\ y_{i,2}\ \cdots\ y_{i,q_i} >$, $2 \leq i \leq t$. Note that in $Y_1$, $a_1 = S[1]$ and $x_1 = D[1]$. If $S[1] = D[1]$, then $Y_1$ contains only $< -x_1 >$. Similar to the routing algorithm working on the cycle structure [6], we can correct the misplaced symbols in $Y_1$ by performing $f_{a_2}, f_{a_3}, \cdots, f_{a_r}, f_{x_1}$. In each other symbol cycle $Y_i$, $2 \leq i \leq t$, we can first correct any symbol in $Y_i$. For instance, assume that the source node is 643512, and the destination node is 425136. Then the $SCS$ is $< 6\ 2\ -4 >< 3\ 1\ 5 >$, and some of the shortest paths are shown as follows.

(a) $(643512) \xrightarrow{f_2} (243516) \xrightarrow{f_4} (423516) \xrightarrow{f_3} (324516) \xrightarrow{f_1} (124536) \xrightarrow{f_5} (524136) \xrightarrow{f_4} (425136)$

(b) $(643512) \xrightarrow{f_2} (243516) \xrightarrow{f_4} (423516) \xrightarrow{f_5} (523416) \xrightarrow{f_3} (325416) \xrightarrow{f_1} (125436) \xrightarrow{f_4} (425136)$

(c) $(643512) \xrightarrow{f_2} (243516) \xrightarrow{f_4} (423516) \xrightarrow{f_1} (123546) \xrightarrow{f_5} (523146) \xrightarrow{f_3} (325146) \xrightarrow{f_4} (425136)$

For the routing path according to the $SCS$, we have to note the following.

(1) The order of $Y_1$, $Y_2$, $\cdots$, $Y_t$ is arbitrary. For the same example, some more shortest paths are shown as follows.

(a) $(643512) \xrightarrow{f_3} (346512) \xrightarrow{f_1} (146532) \xrightarrow{f_5} (546132) \xrightarrow{f_6} (645132)$

$\xrightarrow{f_2}$ (245136) $\xrightarrow{f_4}$ (425136)

(b) (643512) $\xrightarrow{f_5}$ (543612) $\xrightarrow{f_3}$ (345612) $\xrightarrow{f_1}$ (145632) $\xrightarrow{f_6}$ (645132) $\xrightarrow{f_2}$ (245136) $\xrightarrow{f_4}$ (425136)

(c) (643512) $\xrightarrow{f_1}$ (143562) $\xrightarrow{f_5}$ (543162) $\xrightarrow{f_3}$ (345162) $\xrightarrow{f_6}$ (645132) $\xrightarrow{f_2}$ (245136) $\xrightarrow{f_4}$ (425136)

(2) The symbol cycle $Y_i$, $2 \leq i \leq t$, can be nested within any other $Y_j$, $1 \leq j \leq t$, $j \neq i$ (will be explained later.). And the nesting can be done recursively. For example, some other shortest paths are shown as follows.

(a) (643512) $\xrightarrow{f_2}$ (243516) $\xrightarrow{f_3}$ (342516) $\xrightarrow{f_1}$ (142536) $\xrightarrow{f_5}$ (542136) $\xrightarrow{f_2}$ (245136) $\xrightarrow{f_4}$ (425136)

(b) (643512) $\xrightarrow{f_2}$ (243516) $\xrightarrow{f_5}$ (543216) $\xrightarrow{f_3}$ (345216) $\xrightarrow{f_1}$ (145236) $\xrightarrow{f_2}$ (245136) $\xrightarrow{f_4}$ (425136)

(c) (643512) $\xrightarrow{f_2}$ (243516) $\xrightarrow{f_1}$ (143526) $\xrightarrow{f_5}$ (543126) $\xrightarrow{f_3}$ (345126) $\xrightarrow{f_2}$ (245136) $\xrightarrow{f_4}$ (425136)

Next, in the symbol cycle structure, we will see how a structure $p$ changes to $p'$ after a function $f$ is performed. Suppose the source node and the destination node are $S$ and $D$ respectively. We first consider the cases that each $SCS$ contains three symbol cycles as follows.

**Case 1:** $S[1] = D[1]$. And suppose the $SCS$ of $S$ relative to $D$ is $p =< -x_1 >< a_1\ a_2\ \cdots\ a_k >< d >$

**case 1.1:** $f_{a_j}, 1 \leq j \leq k$, is performed.

$p' =< a_j\ a_{j+1}\ a_{j+2}\ \cdots\ a_k\ a_1\ a_2\ \cdots\ a_{j-1}\ -x_1 >< d >$

**case 1.2:** $f_d$ is performed where $d$ is already in its correct position.

$p' =< d\ -x_1 >< a_1\ a_2\ \cdots\ a_k >$

**Case 2:** $S[1] \neq D[1]$. And suppose the $SCS$ of $S$ relative to $D$ is $p =< a_1\ a_2\ \cdots\ a_r\ -x_1 >< b_1\ b_2\ \cdots\ b_q >< d >$

**case 2.1:** $f_{a_2}$ is performed.

$p' =< a_2\ a_3\ \cdots\ a_r\ -x_1 >< b_1\ b_2\ \cdots\ b_q >< d >< a_1 >$

**case 2.2:** $f_{a_j}, 3 \leq j \leq r$, is performed.

$p' =< a_j\ a_{j+1}\ \cdots\ a_r\ -x_1 >< a_1\ a_2\ a_3\ \cdots\ a_{j-1} >< b_1\ b_2\ \cdots\ b_q >< d >$

**case 2.3:** $f_{x_1}$ is performed.

$p' =< -x_1 >< a_1\ a_2\ \cdots\ a_r >< b_1\ b_2\ \cdots\ b_q >< d >$

**case 2.4:** $f_{b_j}, 1 \leq j \leq q$, is performed.

$p' =< b_j\ b_{j+1}\ \cdots\ b_q\ b_1\ b_2\ \cdots\ b_{j-1}\ a_1\ a_2\ \cdots\ a_r\ -x_1 >< d >$

**case 2.5:** $f_d$ is performed where $d$ is already in its correct position.

$p' =< d\ a_1\ a_2\ \cdots\ a_r\ -x_1 >< b_1\ b_2\ \cdots\ b_q >$

Note that the general cases for $SCS$ can be applied similarly. For instance, assume that the source node is 643512, and the destination node is 425136. Then the $SCS$ of $S$ relative to $D$ is $p =< 6\ 2\ -4 >< 3\ 1\ 5 >$, and $f_2(p) =< 2\ -4 >< 3\ 1\ 5 >$, $f_4(p) =< -4 >< 6\ 2 >< 3\ 1\ 5 >$ and $f_5(p) =< 5\ 3\ 1\ 6\ 2\ -4 >$.

Another example is to illustrate the case $S[1] = D[1]$. Assume that the source node is 643512, and the destination node is 645132. Then the $SCS$ of $S$ relative to $D$ is $p =< -6 >< 2 >< 4 >< 3\ 1\ 5 >$, and $f_5(p) =< 5\ 3\ 1\ -6 >< 2 >< 4 >$ and $f_2(p) =< 2\ -6 >< 1\ 5\ 3 >< 4 >$.

We will see how to correct the misplaced symbols in a symbol cycle. Assume that the current node is $(i)*$, and the symbol cycle is $< i\ a_1\ a_2\ \cdots\ a_r\ -x_1 >$. To correct the misplaced symbols, the only routing sequence with minimum length is $f_{a_1},\ f_{a_2},\ \cdots,\ f_{a_r},\ f_{x_1}$ and the corresponding minimal routing path is $(i)*,\ (a_1)*,\ (a_2)*,\ \cdots,\ (a_r)*,\ (x_1)*$. Assume that the symbol cycle is $< a_1\ a_2\ \cdots\ a_r >$, where $a_j \neq i$, $1 \leq j \leq r$. To correct the misplaced symbols, the following routing sequences are all minimal:

$$
\begin{array}{l}
f_{a_1},\ f_{a_2},\ \cdots,\ f_{a_r},\ f_{a_i}, \\
f_{a_2},\ f_{a_3},\ \cdots,\ f_{a_r},\ f_{a_1},\ f_{a_i}, \\
\qquad\qquad \vdots \\
f_{a_r},\ f_{a_1},\ \cdots,\ f_{a_{r-2}},\ f_{a_{r-1}},\ f_{a_i}.
\end{array}
$$

The routing sequence $f_{a_1}, f_{a_2}, \cdots,\ f_{a_r},\ f_{a_i}$ corresponds to the path $(i)*$, $(a_1)*$, $(a_2)*, \cdots,\ (a_r)*, (i)*$. Note both the head and the tail of the routing path are node $(i)*$.

Next, we shall see how a cycle is nested within another cycle. Suppose the current node is $(i)*$, $Y_u =< a_1\ a_2\ \cdots\ a_r >$ and $Y_v =< b_1\ b_2\ \cdots\ b_q >$. And also suppose the sequence $f_{a_1},\ f_{a_2},\ \cdots,\ f_{a_r},\ f_i$ is used to correct the symbols in $Y_u$ and the sequence $[F]$ is used to correct the symbols in $Y_v$. If $Y_v$ is nested within $Y_u$ between $f_{a_j}$ and $f_{a_{j+1}}$, $1 \leq j \leq r-1$, then routing sequence for $Y_u$ and $Y_v$ will be $f_{a_1}, \cdots, f_{a_j}, [F], f_{a_j}, f_{a_{j+1}}, \cdots, f_{a_r}, f_i$ and the routing path is $(i)*, (a_1)*, \cdots, (a_j)*, [Y_v], (a_j)*, (a_{J+1})*, \cdots, (a_r)*, (i)*$. From the above, we can easily give the general nesting rules for the routing sequence. For given a source node $S$ and a destination node $D$, there may be many choices to select the first routing step. A function $f$ is said to be *valid* if it can be applied in the first routing step to get a shortest routing path. When a valid function is applied, the corresponding symbol, which is exchanged with the leftmost symbol, is said to be *valid*.

## 3 A Fully Adaptive Algorithm

In this section, we will propose a fully adaptive wormhole routing algorithm on the star graph. To avoid deadlock and provide adaptive capability, our virtual channel allocation method is as follows.

(1) Initially, the message is routed via the first virtual channel.

(2) Suppose that the message is routing in the $i$th virtual channel now. Only if the polarity of the previous link is negative and the polarity of the next link to be routed is positive, then the message routing is switched to the $(i+1)$th virtual channel of the next link. In other cases, the message is still routed via the $i$th virtual channel.

Our minimal fully adaptive routing algorithm is given as follows.

**Algorithm Minimal Fully Adaptive (MFA)**
**Input:** A source node $S$ and a destination node $D$
**Output:** The routing path with the corresponding virtual channel used in each link.
**Step 1:** Set the current virtual channel number $k = 1$. Set the previous link polarity to positive.
**Step 2:** If $S = D$, then stop; otherwise construct the $SCS$ of the permutation between $S$ and $D$.
**Step 3:** From $SCS$, arbitrarily select a valid function $f$, say $f_b$. Applying the virtual channel allocation method to allocate the next virtual channel of the next link. Then transfer the source message to the next node $f_b(S)$.
**Step 4:** Let $S$ be $f_b(S)$. Go to Step 2.

Before proving Algorithm MFA is deadlock-free, we need the following lemma.

**Lemma 1.** *In a directed circuit on a star graph, there exists at least one link whose polarity is positive and at least one link whose polarity is negative.*

**Proof**: A directed circuit is formed by starting a source node, passing through some intermediate nodes and returning to the source node. We have to consider the following two cases.

Case 1: Assume there exists a circuit, and link polarities in this circuit are all positive. Then the circuit can be represented as a path $P = [(i_1)*, (i_2)*, \cdots, (i_m)*]$, where $i_1 < i_2 < \cdots < i_m$. Obviously, it can not return to the source node. It contradicts the assumption that $P$ is a circuit. In other words, it can not form a circuit if all nodes polarities are positive.

Case 2: Similarly, there is no such circuit with all link polarities being negative.

Thus, this completes the proof. □

**Theorem 2.** *Algorithm MFA is minimal, deadlock-free and fully adaptive.*

**Proof**: Clearly, messages are routed via one of the shortest paths from the source node to the destinate node. So the algorithm is minimal [6]. Since the virtual networks are used in nondecreasing order (the virtual channel identifiers are nondecreasing), it can not cause deadlock between different virtual networks. So, we have to prove only that there is no deadlock in the same virtual network. Assume that Algorithm MFA causes a deadlock in the $i$th virtual network, i.e., there exists a directed circuit in the $i$th virtual network. By Lemma 1, in this circuit there must exist a physical link $l$, whose polarity is positive and whose previous link polarity is negative. Then by Algorithm MFA, if the message is going to be routed via physical link $l$, then it must be routed via the $(i+1)$th virtual channel of link $l$. Therefore, it will not cause a deadlock. And, since there is no specific routing order among the symbol cycles, Algorithm MFA is fully adaptive. □

**Theorem 3.** *In $S_n$, Algorithm MFA needs at most $\lfloor \frac{3n+1}{4} \rfloor$ virtual channels for each physical link.*

**Proof**: Since the message is always routed via one shortest path, we have to consider only the maximum shortest path in $S_n$. The length of the maximum shortest path between two nodes, which is the diameter, is $\lfloor \frac{3(n-1)}{2} \rfloor$ [1]. Algorithm MFA switches to the next virtual channel only when the polarity of the previous link is negative and the polarity of the next link is positive. In Algorithm MFA, the worst case of the polarity sequence is that negative and positive alternate with each other by starting with negative. And the length of the longest polarity sequence is the diameter, i.e., $\lfloor \frac{3(n-1)}{2} \rfloor$. Therefore, the number of virtual channel switchings are at most $\lfloor \frac{3(n-1)}{4} \rfloor$. Algorithm MFA always begins to transmit message by using the first virtual channel. Hence, it needs at most $\lfloor \frac{3(n-1)}{4} \rfloor + 1 = \lfloor \frac{3n+1}{4} \rfloor$ virtual channels for each physical link. □

**Theorem 4.** *Executing Algorithm MFA on $S_n$, there exists one shortest path which needs $\lfloor \frac{3n+1}{4} \rfloor$ virtual channels.*

**Proof**: Four cases have to be considered.

Case 1: $n$ is odd and $S[1] = D[1]$. Assume the source node is of the form $(k)*$, where $k = \frac{n+1}{2}$. From the source node, there exists a destination node such that the $SCS$ is $< -k >< p_1\ p_2 >< p_3\ p_4 > \cdots < p_{n-2}\ p_{n-1} >$ , where $p_i \neq k$, for $1 \leq i \leq n-1$, $p_{4i+1} < k < p_{4i+2}$, for $0 \leq i \leq \lfloor \frac{n-3}{4} \rfloor$, and $p_{4i+3} > k > p_{4i+4}$, for $0 \leq i \leq \lfloor \frac{n-5}{4} \rfloor$. Then one of the shortest routing paths is $(k)*, (p_1)*, (p_2)*, (k)*, (p_3)*$ , $(p_4)*, (k)*, (p_5)*, (p_6)*$ , $(k)*, \cdots (k)*$, $(p_{n-2})*$, $(p_{n-1})*, (k)*$. In the link polarity sequence, negative and positive alternate and its length is $\frac{3(n-1)}{2}$. Thus, $\frac{3(n-1)}{4} + 1 = \lfloor \frac{3n+1}{4} \rfloor$ virtual channels are needed.

Case 2: $n$ is odd and $S[1] \neq D[1]$. In this case, the maximum shortest path is shorter than the maximum shortest path in case 1. Therefore, we do not consider the case.

Case 3: $n$ is even and $S[1] \neq D[1]$. Assume the source node is of the form $(k)*$, where $k = \frac{n}{2}$ *or* $\frac{n}{2} + 1$. From the source node, there exists a destination node such that the $SCS$ is $< k\ -x_1 >< p_1\ p_2 >< p_3\ p_4 > \cdots \cdots < p_{n-3}\ p_{n-2} >$, where $x_1 = D[1]$, $p_i \neq k$, $p_i \neq x_1$, for $1 \leq i \leq n-2$, $p_{4i+1} < k < p_{4i+2}$, for $0 \leq i \leq \lfloor \frac{n-4}{4} \rfloor$, and $p_{4i+3} > k > p_{4i+4}$, for $0 \leq i \leq \lfloor \frac{n-6}{4} \rfloor$. One of the shortest routing paths is $(k)*$, $(p_1)*$, $(p_2)*$, $(k)*$, $(p_3)*$, $(p_4)*$, $(k)*$, $(p_5)*$, $(p_6)*$, $(k)*$, $\cdots$ $(k)*$, $(p_{n-3})*$, $(p_{n-2})*$, $(k)*$, $(x_1)*$. Note the last link polarity from $(k)*$ to $(x_1)*$ may be positive or negative. In the link polarity sequence, negative and positive alternate, except that the last link polarity may not. And the length of the sequence, including the last link, is $\frac{3(n-2)}{2} + 1 = \frac{3n-4}{2}$. Thus, $\frac{3n-4}{4} + 1 = \frac{3n}{4} = \lfloor \frac{3n+1}{4} \rfloor$ virtual channels are needed.

Case 4: $n$ is even and $S[1] = D[1]$. In the maximum shortest path, one of the symbol cycles in $SCS$ must contain three symbols. Similar to case 3, we can correct the symbol cycles one by one, and correct the symbol cycle of three symbols at last. Then in the link polarity sequence, negative and positive alternate, except that the last link polarity may not.

From the above cases, the theorem holds. □

Combining Theorem 3 and Theorem 4, we obtain that in Algorithm MFA,

the number of virtual channels required for each physical link in $S_n$ is exactly $\lfloor \frac{3n+1}{4} \rfloor$.

We use some examples to illustrate Theorem 4. In $S_6$, assume the source node is 465132 and destination node is 123456, then the $SCS$ is $< 4\ -1 >< 3\ 5 >< 2\ 6 >$. One of the shortest routing paths is

$$(465132) \xrightarrow[-]{f_2} (265134) \xrightarrow[+]{f_6} (625134) \xrightarrow[-]{f_4} (425136) \xrightarrow[+]{f_5} (524136)$$
$$\xrightarrow[-]{f_3} (324156) \xrightarrow[+]{f_4} (423156) \xrightarrow[-]{f_1} (123456)$$

which requires four virtual channels.

For example, in $S_7$, assume the source node is 4316752 and destination node is 4561237, then the $SCS$ is $< -4 >< 1\ 6 >< 3\ 5 >< 2\ 7 >$. One of the shortest routing paths is

$$(4316752) \xrightarrow[-]{f_1} (1346752) \xrightarrow[+]{f_6} (6341752) \xrightarrow[-]{f_4} (4361752) \xrightarrow[+]{f_5} (5361742)$$
$$\xrightarrow[-]{f_3} (3561742) \xrightarrow[+]{f_4} (4561732) \xrightarrow[-]{f_2} (2561734) \xrightarrow[+]{f_7} (7561234) \xrightarrow[-]{f_4} (4561237)$$

which requires five virtual channels.

## 4 A Partially Adaptive Algorithm

Though Algorithm MFA is minimal and fully adaptive, it is well known that the more virtual channels are required for each physical channel, the more cost is needed. In this section, we shall reduce the number of virtual channels by making some restrictions on the routing path selection. One *link polarity change* is defined as the link polarity changes from negative to positive or from positive to negative. So, if we can decrease the number of the link polarity changes, then the number of virtual channels can be reduced.

In the star graph, suppose that destination node is $D$. If a symbol cycle $Y$ of an $SCS$ has $k$ symbols and it does not contain symbol $D[1]$, then there are $k$ different orderings to correct the misplaced symbols. Thus, there are $k$ different routing paths corresponding to those orderings. If we make some restrictions on the ordering of routing, then the number of link polarity changes can be reduced. To reduce the number of link polarity changes, we give the following rules. Assume the current node is $(s)*$.

**(R1)** If there exists one valid function $f_t$ such that the link polarity from $(s)*$ to $(t)*$ is the same as previous link polarity, then transfer messages from $(s)*$ to $(t)*$ via $f_t$.

**(R2)** Otherwise, suppose there exists one valid function $f_t$ in a symbol cycle $Y$ such that after $f_t$ is performed, there will be two consecutive links having the same link polarity when other symbols in $Y$ are corrected. Then transfer messages from $(s)*$ to $(t)*$ via $f_t$.

**(R3)** Otherwise, select an arbitrary valid function $f_t$, and then transfer messages from $(s)*$ to $(t)*$ via $f_t$.

For instance, suppose there is a symbol cycle $< 5\ 3 >$ and the current node is $(4)*$. If the previous link polarity is negative, then the routing path is $(i)* \xrightarrow{-} (4)* \xrightarrow{-} (3)* \xrightarrow{+} (5)* \xrightarrow{-} (4)*$. And if the previous link polarity is positive, then the routing path is $(i)* \xrightarrow{+} (4)* \xrightarrow{+} (5)* \xrightarrow{-} (3)* \xrightarrow{+} (4)*$. As another example, suppose the previous link polarity is negative, the current node is $(4)*$, and there is only one symbol cycle $< 5\ 7\ 6 >$ whose symbols are all larger than 4. Then the routing sequence is $(i)* \xrightarrow{-} (4)* \xrightarrow{+} (5)* \xrightarrow{+} (7)* \xrightarrow{-} (6)* \xrightarrow{-} (4)*$, or $(i)* \xrightarrow{-} (4)* \xrightarrow{+} (7)* \xrightarrow{-} (6)* \xrightarrow{-} (5)* \xrightarrow{-} (4)*$, other than $(i)* \xrightarrow{-} (4)* \xrightarrow{+} (6)* \xrightarrow{-} (5)* \xrightarrow{+} (7)* \xrightarrow{-} (4)*$

Our partially adaptive routing algorithm is described as follows.

**Algorithm Minimal Partially Adaptive (MPA)**

**Input:** A source node $S$ and a destination node $D$

**Output:** The routing path with the corresponding virtual channel used in each link.

**Step 1:** Set the current virtual channel number $k = 1$. Set the previous link polarity to positive.

**Step 2:** If $S = D$, then stop; otherwise construct the $SCS$ of the permutation between $S$ and $D$.

**Step 3:** From $SCS$, by rules (R1), (R2) and (R3), select a valid function, say $f_b$. Applying the virtual channel allocation method to allocate the next virtual channel of the next link. Then transfer the source message to the next node $f_b(S)$.

**Step 4:** Let $S$ be $f_b(S)$. Go to Step 2.

**Lemma 5.** *If a symbol cycle $Y$ contains $k$ distinct symbols, then the number of link polarity changes in $Y$ is at most $k$ by applying rules (R1), (R2) and (R3).*

**Proof**: If symbol $D[1]$ is included in $Y$, then it is trivial, since it needs only $k-1$ steps to correct the $k$ misplaced symbols in $Y$. Therefore, we consider the case that symbol $D[1]$ is not included in $Y$. Then we need $k+1$ steps to correct the $k$ misplaced symbols. So, we want to prove that in the $k+1$ steps, there exists two consecutive links such that their link polarities are the same. Assume that the current node is $(s)*$, and $Y =< t_1\ t_2\ \cdots\ t_k >$, where $t_i \neq t_j$ if $i \neq j$, and $t_i \neq s$ for $1 \leq i \leq k$.

Case 1: There exists an integer $i$, such that $t_i < s < t_{(i\ mod\ k)+1}$, where $1 \leq i \leq k$. Since the symbols $t_1,\ t_2,\ \cdots, t_k$ in $Y$ is circular, for convenient representation, we simplify $t_{(i\ mod\ k)+1}$ as $t_{i+1}$. If the previous link polarity is negative, then the routing sequence is $\xrightarrow{-} (s)* \xrightarrow{-} (t_i)* \xrightarrow{+} (t_{i+1})* \longrightarrow \cdots \longrightarrow (t_k)* \longrightarrow (t_1)* \cdots (t_{i-1})* \longrightarrow (s)*$. If the previous link polarity is positive, then we can apply the routing sequence $\xrightarrow{+} (s)* \xrightarrow{+} (t_{i+1})* \longrightarrow (t_{i+2})* \longrightarrow \cdots \longrightarrow (t_k)* \longrightarrow (t_1) * \cdots (t_i)* \longrightarrow (s)*$. To correct the misplaced symbols in $Y$, it needs $k+1$ steps. Since the first link polarity is the same as the previous link polarity, the number of link polarity changes is at most $k$.

Case 2: $s < t_i,\ 1 \leq i \leq k$. If the previous link polarity is positive, then it is the same as case 1. We have to consider only that the previous link polarity is negative. Since the symbols are all distinct, we can find a symbol $t_j$ such that $t_j < t_{j+1}$, where $1 \leq j \leq k-1$. Then we can apply the routing sequence $\stackrel{-}{\longrightarrow}$ $(s)* \stackrel{+}{\longrightarrow} (t_j)* \stackrel{+}{\longrightarrow} (t_{j+1})* \longrightarrow \cdots \longrightarrow (t_k)* \longrightarrow (t_1) * \cdots (t_{j-1})* \longrightarrow (s)*$. Then the number of link polarity changes induced by $Y$ is at most $k$.

Case 3: $s > t_i,\ 1 \leq i \leq k$. It is similar to case 2.

From the above cases, the proof is completed. □

**Lemma 6.** *Suppose two symbol cycles $Y_u$ and $Y_v$, which contain $k_1$ and $k_2$ distinct symbols respectively. Then the number of link polarity changes in $Y_1$ and $Y_2$ is at most $k_1 + k_2$ by applying rules (R1), (R2) and (R3).*

**Proof**: If we first correct $Y_u$ and then $Y_v$, or $Y_v$ and then $Y_u$, then by Lemma 5, the lemma holds. So, we consider the case that one symbol cycle is nested within the other symbol cycle. Without loss of generality, assume that $Y_v$ is nested within $Y_u$. Suppose $Y_v$ is nested between symbols $i$ and $j$ in $Y_u$, then the routing sequence is $\cdots$ , $(i)*$, $[[Y_v]]$, $(i)*]$, $(j)*$, $\cdots$. By Lemma 5, to correct the misplaced symbols in $Y_v$, the number of link polarity changes is at most $k_2$. And the link polarity changes of correcting misplaced symbols in $Y_u$ will not increase after $Y_v$ is nested within $Y_u$, since without nesting, the original previous node of $(j)*$ is $(i)*$, and, with nesting, the previous node of $(j)*$ is still $(i)*$. Thus, the proof is completed. □

**Theorem 7.** *In Algorithm MPA, $\lfloor \frac{n+1}{2} \rfloor$ virtual channels are enough for each physical link in $S_n$.*

In Algorithm MPA, we make some restriction on the routing path selection, but there are still more than one shortest path between the source and the destination. Thus, Algorithm MPA is partially adaptive and minimal.

## 5 A Fault-Tolerant Algorithm

Fault-tolerant routing is an important issue in interconnection networks, because we may not expect that all nodes and links in an interconnection network never fail. An interconnection network is fault-tolerant if all of the non-faulty nodes would be able to communicate with each other. A routing algorithm is said to be *$f$-fault tolerant* if the algorithm is still able to transmit data from the source node to the destination node when it encounters $f$ or fewer faults.

In this section, we will consider that there exists at most one fault on the star graph. And we shall propose a fault-tolerant and partially adaptive routing algorithm. In the previous works, the researchers often proposed a fault-tolerant adaptive routing algorithm by adding extra virtual channels [2, 11]. Our algorithm can tolerate one fault without adding any extra virtual channel. Before giving our algorithm, we have the following assumptions.

(1) A faulty node can not be a source node or a destination node.

(2) The incoming links of a node are considered as a part of the node. If any part of a node fails, then the node fails.

(3) Each node has only the information of its neighbors.

In a star graph, suppose the distance between nodes $v_1$ and $v_3$ is two. Clearly, in a sub-star $S_3$, there are two node disjoint paths from $v_1$ to $v_3$, as shown in Figure 2, one is $v_1 \rightarrow v_2 \rightarrow v_3$ and the other is $v_1 \rightarrow v_6 \rightarrow v_5 \rightarrow v_4 \rightarrow v_3$. We denote the paths of length two and length four in $S_n$ as $P_{two}$ and $P_{four}$ respectively.

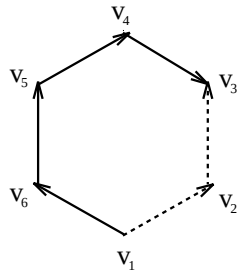


**Fig. 2.** Two node disjoint paths from $v_1$ to $v_3$ in $S_3$.

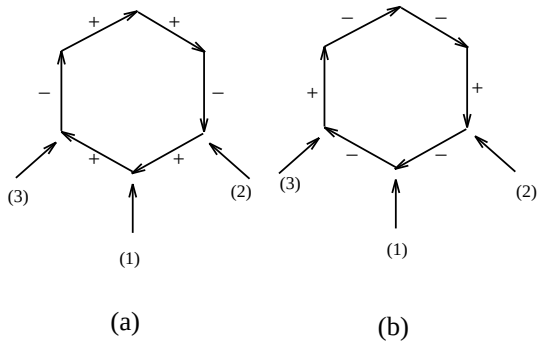


**Fig. 3.** Two kinds of link polarity changes in $S_3$.

**Lemma 8.** *In $S_3$, suppose the distance between nodes $x$ and $y$ is 2. From $x$ to $y$, the number of virtual channels needed in routing through path $P_{four}$ is one more than routing through path $P_{two}$.*

**Proof**: In $S_3$, there are two kinds of link polarity changes, as shown in Figure 3. Assume the direction of the circuit is clockwise. Then we have to consider six

different entering points from other parts of $S_n$. For each entering point, there are two node disjoint paths as shown in Figure 2. Note we also need to consider the link polarities of the incoming link and the outgoing link. By examing each case, the result in this lemma can be obtained. □

**Theorem 9.** *Suppose there is one faulty node in $S_n$, then at most one additional virtual channel is sufficient to make Algorithm MFA and Algorithm MPA one-fault tolerant.*

To tolerate one fault, the routing path may become non-minimal, and it needs one additional virtual channel. To reduce the number of virtual channels, we propose an improved method, which is added with the following rule:

**(R4)** When a fault is encountered and there is no choice other than path $P_{four}$, we do not switch the virtual channel of each link on the $P_{four}$ to other virtual channel.

We will show that it does not need any additional virtual channel and it is still deadlock-free when rule (R4) is applied. Before proving that no deadlock occurs, we give the following lemmas. In the following lemmas, we use $N_1$ to denote the set consisting of the leftmost symbols of all nodes which form a circuit in $S_n$.

**Lemma 10.** *In $N_1$ of a circuit, each symbol appears at least twice in the circuit.*

**Lemma 11.** *For each circuit in $S_n$, the number of link polarity changes from negative to positive is at least twice.*

**Theorem 12.** *Suppose there is one faulty node in $S_n$, Algorithm MFA and Algorithm MPA can tolerate one fault without any extra virtual channel.*

## 6 Conclusion

In a multiprocessor system, processors often need to communicate with each other by message-passing. Because the performance of a multiprocessor system deeply depends on switching techniques and routing methods, a good message routing method is required to achieve low latency and high performance possibly. Wormhole routing becomes popular since it reduces the communication latency and the amount of buffers. Recently, wormhole routing has been widely used in interconnected multiprocessor systems. The routing deadlock can be overcome by using the concept of virtual channels. When a wormhole routing is desinged, one of the most important parameters is the number of virtual channels used per physical link.

In this paper, we define the symbol cycle structure, instead of commonly used cycle structure, to represent the permutation (relation) between two nodes on the star graph. We use another point of views to see the link label. Then with this notation, the link label is also redefined. For two nodes $u$ and $v$ on a star graph, suppose their first (leftmost) symbols are $a$ and $b$ respectively.

The label of the link from $u$ to $v$ is $f_b$ and the label of the link from $v$ to $u$ is $f_a$. The labels of one link on the two directions are not the same. Based upon this label representation and the first symbol of each node, we define our link polarity. And then based on this link polarity, we design our wormhole routing algorithms, which reduce the number of virtual channels for each link.

## References


1. S. B. Akers, D. Horel, and B. Krishnamurthy, "The star graph: An attractive alternative to the n-cube," *Proceeding of the International Conference on Parallel Processing*, pp. 393–400, 1987.
2. S. Chalasani and R. V. Boppana, "Adaptive wormhole routing in tori with faults," *IEE Proceedings Computers and Digital Techniques*, Vol. 142, No. 6, pp. 386–394, Nov. 1995.
3. M. S. Chen and K. G. Shin, "Adaptive fault-tolerant routing in hypercube multicomputers," *IEEE Transactions on Computers*, Vol. 39, No. 12, pp. 1406–1416, Dec. 1990.
4. W. J. Dally, "Virtual-channel flow control," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 3, No. 2, pp. 194–205, Mar. 1992.
5. W. J. Dally and H. Aoki, "Deadlock-free adaptive routing in multicomputer networks using virtual channels," *IEEE Transactions on Parallel and Distributed Systems*, pp. 466–475, Apr. 1993.
6. K. Day and A. Tripathi, "A comparative study of topological properties of hypercubes and star graphs," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 5, No. 1, pp. 31–38, Jan. 1994.
7. J. Duato, "On the design of deadlock-free adaptive routing algorithm for multicomputers: Design methodologies," *Proceeding of the International Conference on Parallel Architectures and Languages Europe*, pp. 391–405, June 1991.
8. J. Duato, "A new theory of deadlock-free adaptive routing in wormhole networks," *IEEE Transactions on Parallel and Distributed Systems*, pp. 1320–1331, Dec. 1993.
9. P. T. Gaughan and S. Yalamanchili, "A family of fault-tolerant routing protocols for direct multiprocessor networks," *IEEE Transactions on Parallel and Distributed Systems*, pp. 482–496, May 1995.
10. X. Lin, P. K. Mckinley, and L. M. Ni, "The message flow model for routing in wormhole-routed networks," *Proceeding of the International Conference on Parallel Processing*, Vol. 1, pp. 1294–1297, Aug. 1993.
11. D. H. Linder and J. C. Harden, "An adaptive and fault tolerant wormhole routing strategy for k-ary n-cubes," *IEEE Transactions on Computers*, Vol. 40, No. 1, pp. 2–12, Jan. 1991.
12. J. Misic and Z. Jovanovic, "Routing function and deadlock avoidance in a star graph interconnection network," *Journal of Parallel and Distributed Computing*, Vol. 22, pp. 216–228, 1994.
13. L. M. Ni and P. K. Mckinley, "A survey of wormhole routing techniques in direct networks," *Computer*, pp. 62–76, Feb. 1993.
14. C. P. Ravikumar and A. M. Goel, "Deadlock-free wormhole routing algorithms for star graph topology," *IEE Proceedings Computers and Digital Techniques*, Vol. 142, No. 6, pp. 395–400, 1995.