Shortest Path Routing and Fault-Tolerant Routing on de Bruijn Networks

Jyh-Wen Mao
Department of Applied Mathematics, National Sun Yat-sen University, Kaohsiung, Taiwan 804

Chang-Biau Yang
Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung, Taiwan 804

In this paper, we study the routing problem for the undirected binary de Bruijn interconnection network. Researchers have never proposed a shortest path routing algorithm on the undirected binary de Bruijn network. We first propose a shortest path routing algorithm, whose time complexity in the binary de Bruijn network of 2^m nodes is $O(m^2)$. Then, based on our shortest path routing algorithm, we propose two fault-tolerant routing schemes. It is assumed that at most one node fails in the network. In our schemes, two node-disjoint paths are found. Our first fault-tolerant routing algorithm guarantees that one of the two paths is the shortest path, and the other is of length at most $m + \log_2 m + 4$. Our second algorithm can find two node-disjoint paths with lengths at most m and $m + 4$, respectively, if the shortest path is not required in the fault-tolerant routing. © 2000 John Wiley & Sons, Inc.

Keywords: interconnection network; de Bruijn graph; routing; fault tolerance

1. INTRODUCTION

In computer networks and distributed systems, performance and fault tolerance are important issues [1–10]. The time delay of a message deeply depends on the number of hops connecting two computers. The communication time between two computers becomes less if the distance between them is reduced. For tolerating one fault on network devices, two disjoint paths are needed in a network. Viewing a network as a graph, desirable features include low degree, small diameter, high connectivity, and minimal increase in the diameter in the presence of faults. From technology considerations, the number of links (degree) of a processor has to be limited or, even better, kept constant.

The *de Bruijn* graph has received much of attention by researchers as a graph model for networks [1–5, 7, 9–11]. The node set of the undirected binary de Bruijn graph $G(2, m)$ consists of all m-bit binary numbers, which has 2^m nodes. Node $v_m v_{m-1} \cdots v_1$ is connected with nodes $v_{m-1} \cdots v_1 v_m$, $v_{m-1} \cdots v_1 \overline{v_m}$, $v_1 v_m \cdots v_2$ and $\overline{v_1} v_m \cdots v_2$, where $\overline{v_1}$ and $\overline{v_m}$ are the complements of v_1 and v_m, respectively. The graph $G(2, m)$, also called the *shift-and-replace graph* [10], has maximum degree 4, minimum degree 2, and diameter m and admits a simple routing algorithm. For example, Figure 1 shows the undirected binary de Bruijn graph of eight nodes. In the directed binary de Bruijn graph, from a node $v_m v_{m-1} \cdots v_1$, there are an arc to node $v_{m-1} \cdots v_1 v_m$ and an arc to node $v_{m-1} \cdots v_1 \overline{v_m}$. In this paper, we shall study the routing problem on the undirected binary de Bruijn network, $G(2, m)$. If we describe the directed one hereafter, we shall state it explicitly.

A major goal in topology design is to be able to tolerate failures with a relatively small performance degradation. de Bruijn graphs are attractive due to the simplicity of the routing messages between two nodes and the capability of fault tolerance. The degree of a de Bruijn graph is bounded. Note that a small diameter implies a low cost for data routing and a bounded degree makes it easy to construct routing algorithms.

The shortest path from a node V to a node W in directed $G(2, m)$ is obtained by determining the longest substring, common to the right/left of V and to the left/right of W. Then, *L-operations/R-operations* are performed to finish this routing process [2, 8]. However, this method cannot always find the shortest path in an undirected $G(2, m)$. In the previous results of fault-tolerant routing on de Bruijn graphs, most researchers focused on the reduction of the increase of the fault diameter. Esfahanian and Hakimi [5] showed that the diameter of $G(K, m)$, and k-ary de Bruijn graph, increases

Received April 1997; accepted November 1999
Correspondence to: C.-B. Yang; e-mail: cbyang@cse.nsysu.edu.tw
Contract grant number: National Center for High-Performance Computing of the Republic of China
Contract grant number: NCHC-86-08-010

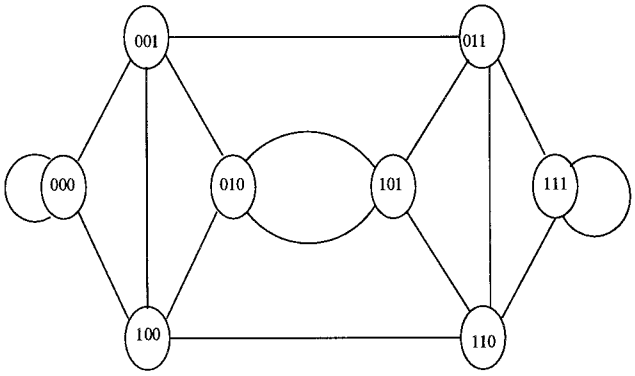

FIG. 1. The 8-node binary de Bruijn graph $G(2,3)$.

by at most $\log_k m + 4$ in the presence of up to $2k-2$ faults. They showed that increase of the fault diameter of $G(k,m)$ grows at most $\log_k \log_\alpha m + 6 + \log_k 5$ when $m \geq 70$, where $k \geq 2$ and α is the golden ratio, that is, $\alpha = (1+\sqrt{5})/2$. In $G(k,m)$, their methods provide $2k-2$ node-disjoint paths with equal length, at most $m + \log_k \log_\alpha m + 6 + \log_k 5$. In these previous results, the node-disjoint paths are all of same length and are not the shortest.

The shortest path routing algorithm for the undirected de Bruijn graph has never been proposed before. In this paper, we shall first propose a shortest path routing algorithm, which requires $O(m^2)$ time. Because of the progress of VLSI technology, the computer components seldom misbehave. When a network is fault-free, it would be more efficient if data are transmitted on a shortest path. Also, when there exists some fault on the network, we may pay some penalty to transmit data on some longer paths. Thus, based on our shortest path routing algorithm, we design a fault-tolerant routing algorithm which provides a shortest path and another node-disjoint path of length at most $m + \log_2 m + 4$. Our algorithm can tolerate one node failure in binary de Bruijn networks and the shortest path can still be used to transmit data between two nodes if no node fails on the path. We also propose another algorithm that can find two node-disjoint paths with lengths at most m and $m+4$, respectively, if the shortest path is not required in the fault-tolerant routing.

The rest of this paper is organized as follows: Section 2 gives some notations used in this paper. In Section 3, we shall propose a shortest path routing algorithm on $G(2,m)$ with time complexity $O(m^2)$. In Section 4, we shall propose two fault-tolerant routing algorithms. Also, finally, some conclusions will be given in Section 5.

2. NOTATIONS

For each node V in $G(2,m)$, let V be identified as an m-bit binary string $v_m v_{m-1} \cdots v_1$. In this paper, we will not distinguish between a node and its identifier string unless stated otherwise. The leftmost bit and the rightmost bit of a binary string X is denoted as X_{left} and X_{right}, respectively. *L-operation* and *R-operation* are used to denote a shift operation on V, whose results are $v_{m-1} \cdots v_1 u$ and $u v_m \cdots v_2$, respectively, where $u = 0$ or 1.

The notation $|X|$ refers to the length of string X. Let $SP(V,W)$ be the shortest path from V to W. It is clear that $|SP(V,W)|$ is at most m since the diameter of $G(2,m)$ is m [8]. An *L-path* and an *R-path* denote the path from V to W by performing only *L-operations* and only *R-operations* on V, respectively. From $V = v_m v_{m-1} \cdots v_1$ to $W = w_m w_{m-1} \cdots w_1$, the *L-path* starting to shift w_i in is denoted as *L-path*(i), which is $v_m v_{m-1} \cdots v_1 \rightarrow v_{m-1} \cdots v_1 w_i \rightarrow v_{m-2} \cdots v_1 w_i w_{i-1} \rightarrow \cdots \rightarrow v_{m-i} \cdots v_1 w_i \cdots w_1$. Similarly, *R-path*$(i) = v_m v_{m-1} \cdots v_1 \rightarrow w_i v_m \cdots v_2 \rightarrow w_{i+1} w_i v_m \cdots v_3 \rightarrow \cdots \rightarrow w_m \cdots w_i v_m \cdots v_{m-i+2}$. We will use X' and $'X$ to denote the rightmost and leftmost substring, with arbitrary length, of X, respectively. Note that a string or a substring may be empty.

3. THE SHORTEST PATH ROUTING ALGORITHM

The diameter of binary de Bruijn graph $G(2,m)$ was proved to be m by Pradhan and Reddy [8]. A simple routing method is as follows: Let $V = v_m v_{m-1} \cdots v_1$ be the source node and $W = w_m w_{m-1} \cdots w_1$ be the destination node. The routing path is $v_m v_{m-1} \cdots v_1 \rightarrow v_{m-1} \cdots v_1 w_m \rightarrow v_{m-2} \cdots v_1 w_m w_{m-1} \rightarrow \cdots \rightarrow v_1 w_m \cdots w_2 \rightarrow w_m w_{m-1} \cdots w_1$. For example, let $V =$ 11100 and $W =$ 10011, the routing path is 11100 $\rightarrow$ 11001 $\rightarrow$ 10010 $\rightarrow$ 00100 $\rightarrow$ 01001 $\rightarrow$ 10011. This path is the *L-path*(m) since it begins at w_m and takes m *L-operations*. We can also establish the *R-path*(1) from 11100 to 10011 by performing m *R-operations*. By the behavior of routing operations in the above example, there exists another path, 11100 $\rightarrow$ 11001 $\rightarrow$ 10011, which requires only two *L-operations*. This is due to the substring 100 being both the rightmost substring of V and the leftmost substring of W. We say that substring 100 is the *L-string* on the *L-path*. The length of the routing path, that is, the *L-path*(2), is reduced by the length of the *L-string*, which is $5-3=2$. Similarly, there exists an *R-string* 11 on the *R-path* which saves two *R-operations*.

Pradhan and Reddy [8] proposed a routing method based on the comparison of the lengths of *L-string* and *R-string*. If the length of the *L-string* is greater than that of the *R-string*, then *L-operations* are performed to finish this routing process; otherwise, *R-operations* are performed. For example, in the above example, on the *L-path*, only two *L-operations* are needed.

It is not difficult to give a counterexample to show that Pradhan and Reddy's routing method could not always find the shortest path from V to W for any V and W under undirected de Bruijn graphs. Suppose that the source node is $V =$ 1011011 and the destination node is

$W = 1011010$. Thus, the *L-string* and *R-string* are 1011 and 10, respectively. It takes three *L-operations* for the routing path by Pradhan and Reddy's algorithm. However, it is easy to show that $1011011 \rightarrow 0101101 \rightarrow 1011010$ is the shortest path from V to W, and it takes only two steps. With this example, we conclude that we may not get the shortest path for a pair of nodes by only considering the *L-string* and *R-string*.

Let V and W be the source and destination nodes, respectively. Suppose that X is a common substring in V and W, where X may be empty. V and W can be represented as $V_L \cdot X \cdot V_R$ and $W_L \cdot X \cdot W_R$, respectively, where a dot means a string concatenation operation. Note that each of V_L, V_R, W_L, and W_R may be empty. Thus, the routing process from V to W can be viewed as to transform the binary representation of V into that of W by using minimal number of *L-operations* and *R-operations*. One possible method to achieve the transformation is to start with $|V_R|$ *R-operations*, then perform $|W_R|$ *L-operations* to correct $|W_R|$ bits on the right side of X in W, then perform $|W_L|$ *L-operations*, and, finally, perform $|W_L|$ *R-operations* to finish this routing process. Let A, B, C, and D be some arbitrary strings of proper lengths and the path of the above process is $V_L \cdot X \cdot V_R \rightarrow \cdots \rightarrow A \cdot V_L \cdot X \rightarrow \cdots \rightarrow (A \cdot V_L)' \cdot X \cdot W_R \rightarrow \cdots \rightarrow X \cdot W_R \cdot B \rightarrow \cdots \rightarrow W_L \cdot X \cdot W_R$. The total number of steps is $|V_R| + |W_R| + |W_L| + |W_L| = m - |X| + |V_R| + |W_L|$. By the behavior of this routing process, the routing path is called an *RLR-path*.

Similarly, another path is $V_L \cdot X \cdot V_R \rightarrow \cdots \rightarrow X \cdot V_R \cdot C \rightarrow \cdots \rightarrow W_L \cdot X \cdot' (V_R \cdot C) \rightarrow \cdots \rightarrow D \cdot W_L \cdot X \rightarrow \cdots \rightarrow W_L \cdot X \cdot W_R$. The path is called an *LRL-path*, with length $(m - |X|) + |V_L| + |W_R|$. Thus, based on a certain substring X, the length of the shortest path from V to W is $(m - |X|) - \min(|V_L| + |W_R|, |V_R| + |W_L|)$. It is clear that each of all possible shortest paths from V to W corresponds to some X. If we examine all possible X's, then we can find a shortest routing path from V to W with length $(m - |X|) + \min(|V_L| + |W_R|, |V_R| + |W_L|)$. Since the diameter of $G(2, m)$ is m, $(m - |X|) + \min(|V_L| + |W_R|, |V_R| + |W_L|) \leq m$ always holds.

There is a simple method to decide which X causes a minimal value of $m - |X| + \min(|V_L| + |W_R|, |V_R| + |W_L|)$. By comparing $w_m w_{m-1} \cdots w_1$ with one substring $v_i v_{i-1} \cdots v_1$ of $V, 1 \leq i \leq m$, or $v_m v_{m-1} \cdots v_1$ with one substring $w_i w_{i-1} \cdots w_1$ of $W, 1 \leq i \leq m$, one best common string X can be found, which needs $O(m)$ time. The desired X can be obtained by examining every possible comparison case. Thus, this method requires $O(m^2)$ time. We summarize the above shortest routing algorithm as follows:

Algorithm 1 *Shortest Routing*

Input: Source node V and destination node W in $G(2, m)$.

Output: A shortest routing path from V to W.

STEP 1. Find the common substring X of V and W such that $m - |X| + \min(|V_L| + |W_R|, |V_R| + |W_L|)$ is minimal.
STEP 2. If $|V_L| + |W_R| \leq |V_R| + |W_L|$ then return *LRL-path* else return *RLR-path*
end

In the algorithm, we have to examine each of all possible X's and the corresponding values of $|V_L|, |V_R|, |W_L|$, and $|W_R|$. Thus, the time complexity is $O(m^2)$ in $G(2, m)$.

Theorem 1. *Algorithm* 1 *finds a shortest path from a source node V to another destination node W in* $G(2, m)$.

Proof. The routing path can be viewed as the process of transforming the bit string representation of V to that of W. The routing method can be achieved by shifting the node's binary representation left or right. It is clear that for the routing data from V to W we have to perform some *L-operations* or *R-operations* from V, via some intermediate nodes, to W. By the transformation process, there may be a bit substring of V which is moved to some bit position of W. In other words, that bit substring is not shifted out.

Suppose that at least two nonempty bit substrings, separated by at least one bit, in V are not shifted out. Let X_1, X_2 be two such substrings. Clearly, the bits between X_1 and X_2 cannot be shifted out. Then, they cannot be corrected to the bits in W, and we cannot route data from V to W. Thus, at most one nonempty substring X in V is not shifted out.

It is clear that V and W can be represented as $V_L \cdot X \cdot V_R$ and $W_L \cdot Y \cdot W_R$, respectively. Also, the minimal number of steps of the substitution of W_L for V_L is to shift out V_L straightly by performing $|V_L|$ *L-operations* and then to shift W_L to the left side of X by performing $|W_L|$ *R-operations* immediately. Moreover, it takes only $2|W_R|$ steps to perform the substitution of W_R for V_R. Thus, the total steps is $|V_L| + |W_L| + 2|W_R| = m - |X| + |V_L| + |W_R|$. In another situation, if we replace V_R by W_R first, the total number of steps will be $m - |X| + |V_R| + |W_L|$. For a certain X, the shortest path is determined by $\min(|V_L| + |W_R|, |V_R| + |W_L|)$.

Let $Num(X)$ be the number of steps that for routing data from V to W while substring X in V is not shifted out. Thus, the length of the shortest path from V to W is the minimal value of $Num(X)$ for all possible X's. This completes the proof. ∎

4. FAULT-TOLERANT ROUTING

In this section, we focus on the fault-tolerant routing method whose goal is to find a shortest path and another node-disjoint path. Due to the topology of de Bruijn networks, there exists an easy way to construct two node-disjoint paths for two adjacent nodes. Thus, we do not

discuss the node-disjoint paths for two adjacent nodes. Because of the structures of source node V and destination node W, the existence of V_L, V_R, W_L, and W_R will affect the path node-disjoint to $SP(V,W)$. Without losing generality, we assume that the shortest path $SP(V,W)$ is an *LRL-path*, that is, $V_{L_{right}} \neq W_{L_{right}}, V_{R_{left}} \neq W_{R_{left}}, m - |X| + |V_L| + |W_R| \leq m$ and $|V_L| + |W_R| \leq |V_R| + |W_L|$. Besides, we have $|V_L| + |V_R| = |W_R| + |W_L|$. Thus, $|V_L| \leq |W_L|, |W_R| \leq |V_R|$, and $|X| \geq |V_L| + |W_R|$. There are seven possible cases to be considered:

1. $|X| \neq 0, |V_L| \neq 0, |V_R| \neq 0, |W_L| \neq 0, |W_R| \neq 0.$
2. $|X| \neq 0, |V_L| \neq 0, |V_R| \neq 0, |W_L| \neq 0, |W_R| = 0.$
3. $|X| \neq 0, |V_L| \neq 0, |V_R| = 0, |W_L| \neq 0, |W_R| = 0$
4. $|X| \neq 0, |V_L| = 0, |V_R| \neq 0, |W_L| \neq 0, |W_R| = 0.$
5. $|X| = 0.$
6. $|X| \neq 0, |V_L| = 0, |V_R| \neq 0, |W_L| \neq 0, |W_R| \neq 0.$
7. $|X| \neq 0, |V_L| = 0, |V_R| \neq 0, |W_L| = 0, |W_R| \neq 0$

By symmetry, Cases 6 and 7 are similar to Cases 2 and 3, respectively. Thus, we will not discuss them.

4.1. Case 1

In this case, the shortest path $SP(V,W)$ is an *LRL-path*. Let $SP(V,W)$ be the following path: $V = V_L \cdot X \cdot V_R \rightarrow \cdots \rightarrow X \cdot V_R \cdot C \rightarrow \cdots \rightarrow W_L \cdot X \cdot' V_R \rightarrow \cdots \rightarrow D \cdot W_L \cdot X \rightarrow \cdots \rightarrow W_L \cdot X \cdot W_R = W$, where C and D are arbitrary strings to be shifted in. If C and D are chosen properly, then we can find another path from V to W which is node-disjoint to $SP(V,W)$.

Lemma 1. *There exist C, D, E, and F such that paths $V = V_L \cdot X \cdot V_R \rightarrow \cdots \rightarrow X \cdot V_R \cdot' E \rightarrow \cdots \rightarrow F \cdot X \rightarrow \cdots \rightarrow F' \cdot X \cdot W_R \rightarrow \cdots \rightarrow X \cdot W_R \cdot E \rightarrow \cdots \rightarrow W_L \cdot X \cdot W_R = W$ and $SP(V,W)$ are node-disjoint.*

Proof. Let every bit of C and D be X_{right} and $W_{L_{right}}$, respectively, and every bit of E and F be $\overline{X_{right}}$ and $\overline{W_{L_{right}}}$, respectively.

The two paths are divided into P_1, P_2 and P_3, P_4, respectively, as shown in Figure 2. We will show that no node appears in both P_i and P_j, where $i = 1$ or 2 and $j = 3$ or 4. By the setting of C, D, E, and F, all possible cases that cause these two paths to share a node are shown in Figure 3. In the figure, each dashed line represents one pair of possible matching positions between two strings. If one position of one string is matched with two or more positions in another string, it just means that we have to consider these matching cases. In Figure 3, Cases (a)–(d) illustrate the comparison of one node in P_1 and one node in P_3. Also, Case (e), Case (f), and Case (g) illustrate the comparison pairs $(P_1, P_4), (P_2, P_3)$, and (P_2, P_4), respectively. In each of the following cases, we first suppose that the equality of the comparison of two nodes holds; then, a contradiction would be obtained.

CASE (a). It implies every bit of X is equal to $\overline{W_{L_{right}}}$ since $F = (\overline{W_{L_{right}}})^*$, where * means that the bit is replicated

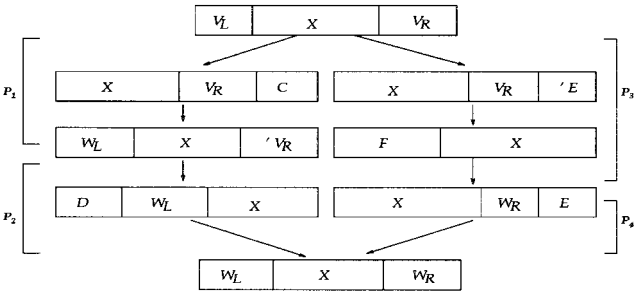

FIG. 2. Two node-disjoint paths in Case 1.

zero or more times. Then, we have $X = (V_{L_{right}})^*$ since $V_{L_{right}} \neq W_{L_{right}}$. Thus, we would find a Y whose content is equal to X and whose position starts at the left position of X in V such that there exists another path with length $m - |Y| + |V_L| - 1 + |W_R| = m - |X| + |V_L| + W_R| - 1$, which is shorter than $SP(V,W)$. It is a contradiction.

CASE (b). The equality does not hold in position 1 by the setting of F. In position 2, we would find a $Y = W'_L \cdot X \cdot' V_R$ such that $m - |Y| + |V_L| + |W_R| < m - |X| + |V_L| + |W_R|$. It is a contradiction.

CASE (c). In position 1, it is similar to position 2 of Case (b). In position 2, the equality does not hold by the setting of E.

CASE (d). In position 1, it is similar to Case (a). The equality does not hold in position 2 since $W_{R_{left}} \neq V_{R_{left}}$. In position 3, we would find a $Y = X' \cdot' W_R$, whose content is equal to X, such that $m - |Y| + |V_L| + |W'_R| < m - |X| + |V_L| + |W_R|$. It is a contradiction.

CASE (e). In position 1, it is similar to position 2 of Case (b). The analysis of position 2 is similar to position 3 of Case (d). Position 3 is similar to position 2 of Case (c).

CASE (f). Position 1 is similar to position 2 of Case (c). The analysis of position 2-1 is the same as that of position 2 of Case (b). In position 2-2, $W_L \cdot X$ is a substring of $X \cdot V_R$. Thus, we found find a $Y = W_L \cdot X$ such that $m - |Y| + |V'_R| < m - |X| + |V_L| + |W_R|$. It is a contradiction.

CASE (g). Positions 1 and 2 are similar to position 2 of Case (c) and position 3 of Case (d), respectively.

This completes the proof. ∎

The length of the path node-disjoint to $SP(V,W)$ in Lemma 1 is $2(m - |X|) + |V_L| + |W_L|$. If $|X|$ is small enough and $|W_L|$ is large enough, then $2(m-|X|)+|V_L|+|W_L|$ is very close to $3m$ since $m-|X|+|V_L|+|W_R| \leq m$. In this situation, it is not a good solution for fault-tolerant routing. Another way to obtain a path node-disjoint to $SP(V,W)$ is to perform only *R-operations* from V to W. Before the bits of W are shifted in, a string ϕ is first shifted in. We will construct a string ϕ such that each substring, except V and W themselves, of string $W \cdot \phi \cdot V$

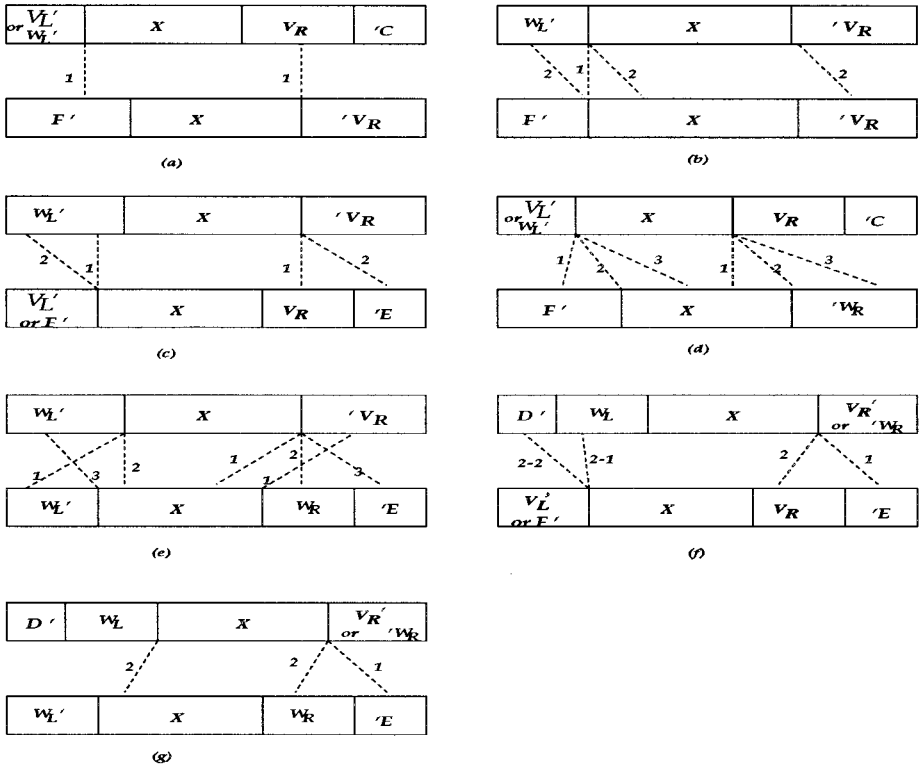

FIG. 3. The comparison of two nodes in two paths.

with length m is not equal to each node in $SP(V, W)$. In other words, $W \cdot \phi \cdot V$ has not a common substring of length m, except V and W, with strings $V_L \cdot X \cdot V_R \cdot C, D \cdot W_L \cdot X \cdot V_R \cdot C$ and $D \cdot W_L \cdot X \cdot W_R$.

There are at most $m + |V_L| - k + 1$ distinct substrings of length k in string $V_L \cdot X \cdot V_R \cdot C$, where $|C| = |V_L|$. Because strings $D \cdot W_L \cdot X \cdot V_R \cdot C$ and $V_L \cdot X \cdot V_R \cdot C$ have a common substring $X \cdot V_R \cdot C$, the number of substrings of length k in string $D \cdot W_L X \cdot V_R \cdot C$ is $|W_R| + |W_L|$ more than that of string $X \cdot V_R \cdot C$, where $|D| = |W_R|$. We conclude that there at at most $(m + |V_L| - k + 1) + (|W_L| + |W_R|) + |W_R| = m + m - |X| + |V_L| + |W_R| - k + 1 \leq 2m - k + 1$ different substrings of length k in strings $V_L \cdot X \cdot V_R \cdot C, D \cdot W_L \cdot X \cdot V_R \cdot C$, and $D \cdot W_L \cdot X \cdot W_R$. Thus, we can find a string δ of length $\log_2 2m$ such that δ is not a substring of strings $V_L \cdot X \cdot V_R \cdot C, D \cdot W_L \cdot X \cdot V_R \cdot C$ and $D \cdot W_L \cdot X \cdot W_R$. We then construct ϕ based upon δ. We require that strings $\phi' \cdot' V$ and $W' \cdot' \phi$ of length m are not substrings of strings $V_L \cdot X \cdot V_R \cdot C, D \cdot W_L \cdot X \cdot V_R \cdot C$ and $D \cdot W_L X \cdot W_R$. Because we cannot make sure that each of $\delta' \cdot' V$ and $W' \cdot' \delta$ with length m is not a substring of strings $V_L \cdot X \cdot V_R \cdot C, D \cdot W_L \cdot X \cdot V_R \cdot C$ and $D \cdot W_L \cdot X \cdot W_R$, we cannot set $\phi = \delta$ simply. A way to construct ϕ based upon δ is to test whether V and W have a *period* or not. A string S is a *period* of a string R if R is a prefix of string S^i(i repetitions of the string S) for some integer $i \geq 1$. Figure 4 illustrates how to construct ϕ, which is obtained by adding at most one bit on the left side of δ and at most one bit on the right side of δ. Initially, ϕ is set to δ. If $V \cdot C$ has a period of length $k, k < |V_L| + \log_2 2m$, then we add $\overline{v_{m-k+1}}$ to the right side of ϕ. If $D \cdot W$ has a period of length $k, k < |W_R| + \log_2 2m$, then we add $\overline{w_k}$ to the left side of ϕ. Thus, the length of ϕ is at most $|\delta| + 2 = \log_2 2m + 2 = \log_2 m + 3$ in the worst case. In other words, the two node-disjoint paths from V to W are $SP(V, W)$ and another path of length at most $m + \log_2 m + 3$.

4.2. *Case 2*

In this case, there is a small difference from Case 1, that is, W_R degrades to an empty string. Thus, $SP(V, W)$ degrades to an *LR-path*, which is $V = V_L \cdot X \cdot V_R \rightarrow \cdots \rightarrow X \cdot V_R \cdot C \rightarrow \cdots \rightarrow W'_L \cdot X \cdot V_R \rightarrow \cdots \rightarrow W_L \cdot X = W$, where C is an arbitrary string to be shifted in. The path that is node-disjoint to $SP(V, W)$ can be obtained by slightly modifying the result in Case 1.

Lemma 2. *There exist C, E, and F such that paths $V = V_L \cdot X \cdot V_R \rightarrow \cdots \rightarrow E \cdot V_L \cdot X \rightarrow \cdots \rightarrow X \cdot F \rightarrow \cdots \rightarrow W_L \cdot X = W$ and $SP(V, W)$ are node-disjoint.*

Proof. Let every bit of C, E, and F be $V_{R_{left}}, \overline{X_{left}}$, and $\overline{V_{R_{left}}}$, respectively. The two paths are divided into P_1, P_2 and P_3, P_4, respectively, as shown in Figure 5. We will show that no node appears in both P_i and P_j, where $i = 1$ or 2 and $j = 3$ or 4.

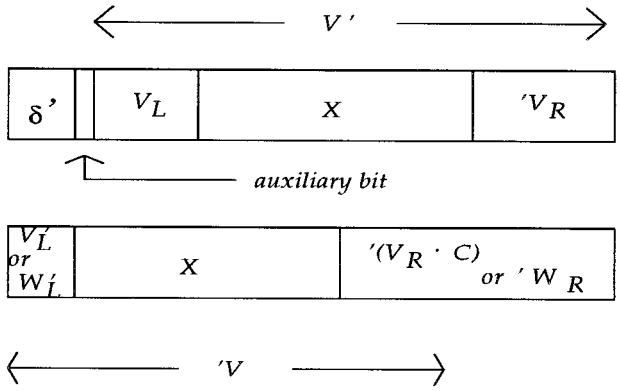

FIG. 4. Construction of ϕ based upon δ.

By the setting of C, E, and F, all possible cases that cause these two paths to share a node are shown in Figure 6, in which the numbers are possible matching positions between two strings. In Figure 6, the comparison of one node in P_1 and one node in P_4 is omitted due to the settings of C and F. Also, Case (c) and Case (d) illustrate the comparison of one node in P_2 and one node in P_4, and Case (a) and Case (b) illustrate the comparison pairs (P_1, P_3) and (P_2, P_3), respectively. The proof of each case is similar to that of Lemma 1. ∎

The lengths of these two paths are $m - |X| + |V_L|$ and $m - |X| + |V_L| + 2|V_R|$, respectively. $m - |X| + |V_L| + 2|V_R|$ is close to $3m$, which $|X|$ is small and $|V_R|$ is large enough. Another way to achieve fault-tolerant routing is similar to the method of Case 1. We construct a string ϕ such that path $V \to \cdots \to \phi' \cdot' V \to \cdots \to W' \cdot \phi \cdot' V \to \cdots \to W' \cdot' \phi \to \cdots \to W$ and $SP(V, W)$ have no common intermediate node. The method to construct ϕ is similar to that of Case 1. Since $|W_R| = 0, \phi_{left}$ should be set to $\overline{V_{R_{left}}}$. Thus, the length of the above path is that of the path in Case 1 plus one, which is at most $m + \log_2 m + 4$ in the worst case. However, in this case, there exists two exceptions: (1) $X = tt\cdots t$ and (2) $W = \bar{u}u\cdots u$ or $uu\cdots u$. For the first case, it implies that $V_{L_{right}} = \bar{t}, W_{L_{right}} = t$ and $V_{R_{left}} = \bar{t}$. Thus, ϕ can be set to $t\bar{t}\overline{V_{R_{left-1}}}\bar{t}$. For the second case, a slight modification on $SP(V, W)$ and a magic direct routing from V to W can solve the routing problem while W is $\bar{u}u\cdots u$ or $uu\cdots u$. $SP(V, W)$ would be $V = V_L \cdot X \cdot V_R \to \cdots \to X \cdot V_R \to \bar{u}\cdots\bar{u} \to \cdots \to W'_L \cdot X \cdot V_R \to \cdots \to W_L \cdot X = W$ and another path would be $V = V_L \cdot X \cdot V_R \to \cdots \to X \cdot V_R \cdot u\cdots u \to \cdots \to V_R \cdot u\cdots u \to \cdots \to W$, which is an *L-path* and whose length is less than or equal to m. Since $V_{R_{left}} = \bar{u}$ and $X = uu\cdots u$ in this case, it is easy to show that the above two paths are node-disjoint, while $|V_R| \leq m/2$. Also, it is not hard to show that the argument also holds while $|V_R| > m/2$.

In this case, we present two node-disjoint paths from V to W which are $SP(V, W)$ and another path of length at most $m + \log_2 m + 4$.

4.3. Case 3

In this case, two node-disjoint shortest paths from V to W can be constructed easily. They are $V = V_L \cdot X \to \cdots \to X \cdot C \to \cdots \to W_L \cdot X = W$ and $V = V_L \cdot X \to \cdots \to X \cdot \bar{C} \to \cdots \to W_L \cdot X = W$, where C is an arbitrary string. We conclude that two node-disjoint paths can be found in this case and both of their lengths are equal to $2(m - |X|)$, which are less than m and the shortest.

4.4. Case 4

In this case, there exists a nonempty string X that is both a leftmost substring of V and a rightmost substring of W. Thus, V and W can be represented as $X \cdot V_R$ and $W_L \cdot X$, respectively. Also, the shortest path from V to W can be represented as $X \cdot V_R \to \cdots \to W'_L \cdot X \cdot' V_R \to \cdots \to W_L \cdot X$, that is, $SP(V, W)$ degrades to *R-path*$(|X| + 1)$. We claim that *L-path*(m) or *R-path*(1) is not always node-disjoint to $SP(V, W)$ for any V, W. It is easy to take an example to show that above claim is correct. Let $V = 101101001$ and $W = 111110110$. Thus, $X = 10110, V_R = 1001$, and $W_L = 1111$. Let $S = W'_L \cdot X \cdot' V_R$, an intermediate node on path $SP(V, W)$ with $|W'_L| = 2$ and $|'V_R| = 2$. Therefore, $S = 111011010$ is also an intermediate node on *R-path*(1). We can conclude that if $'X =' V_R$, then there exists a node shared by both of paths $SP(V, W)$ and *R-path*(1). Another example is $V = 100101$ and $W = 001001$. Thus, $X = 1001, V_R = 01$, and $W_L = 00$. There exists a node $S = 010010$ shared by paths $SP(V, W)$ and *L-path*(m). We will eliminate the effect of V_R and W_L and then give a rotating path from V to W that is node-disjoint to $SP(V, W)$.

Lemma 3. *There exists strings E and F such that $V = X \cdot V_R \to \cdots \to E \cdot X \to \cdots \to X \cdot F \to \cdots \to W_L \cdot X = W$ and $SP(V, W)$ are node-disjoint.*

Proof. Let every bit of E and F be $\overline{W_{L_{right}}}$ and $\overline{V_{R_{left}}}$, respectively. The proof is similar to that of Lemma 1. ∎

The length of the path described in Lemma 3 is $3(m - |X|)$, which is triple of the length of $SP(V, W)$. If $|X|$ is small enough, then $3(m - |X|)$ is close to $3m$. In this situation, it is not a good solution for fault-tolerant

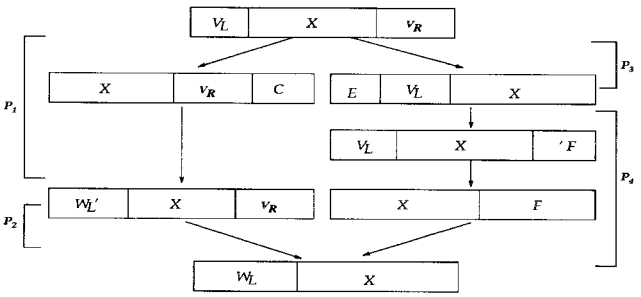

FIG. 5. Two node-disjoint paths in Case 2.

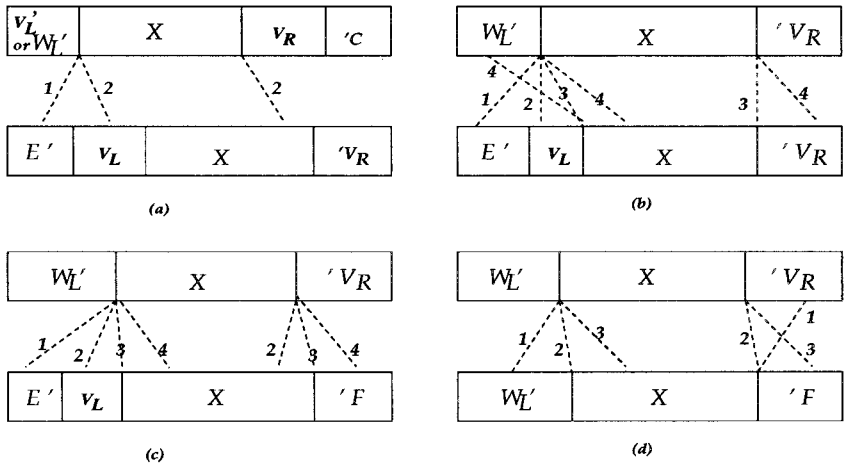

FIG. 6. Comparison of two nodes in two paths.

routing from V to W. Another solution is similar to Sridhar's method [10], but the increase in the diameter of our solution in this case is at most 4. Our solution is similar to Case 1. Only *R-operations* from V to W are performed and a string ϕ is shifted in before W is done, where $V \neq 1\cdots 1u$ or $0\cdots 0u$ and $W \neq t1\cdots 1$ or $t0\cdots 0, u, t = 0$ or 1. In other words, we will construct a string ϕ, $|\phi| \leq 4$, such that $W_L{\cdot}X{\cdot}V_R$ and $W_L{\cdot}X{\cdot}\phi{\cdot}X{\cdot}V_R$ have no common substring of length m except V and W themselves. It is clear that each substring of length m in $W_L{\cdot}X{\cdot}\phi{\cdot}X{\cdot}V_R$ is of form $W'_L{\cdot}X{\cdot}'\phi, W'{\cdot}\phi{\cdot}'V$ or $\phi'{\cdot}X{\cdot}'V_R$. Also, each substring of length m in $W_L \cdot X \cdot V_R$ is only of form $W'_L \cdot X \cdot' V_R$. A way to construct ϕ is to compare each substring of length m in $W_L \cdot X \cdot \phi \cdot X \cdot V_R$ with each substring of length m in $W_L \cdot X \cdot V_R$. The following algorithm shows how to construct ϕ :

Algorithm 2. *Construction of* ϕ

Input: $W_L \cdot X \cdot V_R$, where $W_L = b_s b_{s-1}\cdots b_1$ and $V_R = c_q c_{q-1}\cdots c_1$.

Output: ϕ.

$k = |X|, i = \lfloor k/2 \rfloor$.
$d = 0$ or 1.
$\phi = \overline{c_q} d \overline{b_1}$.
if k is even then $f = 0$ else $f = 1$.
if $X = u^k$ (k repetitions of u), $u = 0$ or 1, then
  if $(b_1 = \bar{u})$ and $(c_q = \bar{u})$ then $\phi = u$.
  if $(c_q = u)$ and $(b_1 = \bar{u})$ then $\phi = \overline{c_q}\overline{b_2}\bar{u}\overline{b_1}$.
  if $(b_1 = u)$ and $(c_q = \bar{u})$ then $\phi = \overline{c_q}\bar{u}\overline{c_{q-1}}\overline{b_1}$.
else
  if X has a period of length $i - j, -1 \leq j \leq i - 1$, **then**
    if $x_{i-j} = c_q$ then $\phi = \overline{c_q x_{i+j+f+2}}\overline{b_1}$
    else if $x_{i+j+f+1} = b_1$ then $\phi = \overline{c_q x_{i-j-1}}\overline{b_1}$.
end

Lemma 4. *Let* ϕ *be the string constructed by Algorithm 2. Strings* $W_L{\cdot}X{\cdot}V_R$ *and* $W_L{\cdot}X{\cdot}\phi{\cdot}X{\cdot}V_R$ *have no common substring of length* m *except* V *and* W.

Proof. The proof is not hard for $X = u^k$ or X has a period length $i - j, 0 \leq j < i$. We now show that Algorithm 2 is correct while X has a period of length $i + 1$. Suppose that k is odd. Thus, there are two comparison patterns in this case. One is the bitwise comparison on $\phi \cdot' X$ and X', that is, the comparison pairs are (ϕ_3, x_{i+3}), (ϕ_2, x_{i+2}), (ϕ_1, x_{i+1}), $(x_k, x_i), \cdots, (x_{i+2}, x_1)$, $(x_{i+1}, V_{R_{left}})$. The other one is the bitwise comparison on $X' \cdot \phi$ and $'X$, that is, the comparison pairs are $(\phi_1, x_{i-1}), (\phi_2, x_i)$, (ϕ_3, x_{i+1}), $(x_1, x_{i+2}), \cdots, (x_i, x_k)$, $(x_{i+1}, W_{L_{right}})$. Thus, if $\phi_2 = \overline{x_{i+2}}$ or $\phi_2 = \overline{x_2}$, then there does not exist a common substring of length m between $W_L \cdot X \cdot V_R$ and $W_L \cdot X \cdot \phi \cdot X \cdot V_R$ except V and W. The arguments also hold while k is even. ∎

By the above lemma, we provide two node-disjoint paths whose lengths are $|SP(V, W)|$ and $m + 4$. Now, we will discuss the node-disjoint paths for the exception nodes, that is, $V = 11\cdots 1u$ or $00\cdots 0u$, or $W = t1\cdots 1$ or $t0\cdots 0$, where $u, t = 0$ or 1. If both V and W are exception nodes, then there exists a simple routing method to achieve the node-disjoint requirement. Otherwise, without losing generality, let $V = u\cdots u\bar{u}$ or $V = u\cdots uu$ if only one of them is an exception node. For $V = u\cdots u\bar{u}$, we can solve this problem by using a simple trick on V. Let $P = u\cdots u\overline{uuu}$ and $\phi = \overline{uw_{l_2}}\bar{u}u$. Then, the routing path from P to W is following the string $W \cdot \phi \cdot P (= W_L{\cdot}u\cdots u{\cdot}\phi{\cdot}u\cdots u\overline{uuu})$. Clearly, V is the next node of P on the path. The lengths of the two node-disjoint paths from V to W are $|SP(P, W)| + 1 = m - |X| + 1 = |SP(V, W)| + 1$ and $m + |\phi| - 1 = m + 3$. For $V = u\cdots uu, V$ is also the neighbor of $u\cdots u\bar{u}$ and $\bar{u}u\cdots u$. Thus, the total lengths of these two node-disjoint paths becomes $|SP(P, W)| + 2 = m - |X| + 2 = |SP(V, W)| + 2$ and $m + |\phi| - 2 + 1 = m + 3$, respectively.

4.5. Case 5

In this case, $|X| = 0$ implies that $|V_L| = 0$ and $|W_R| = 0$. Thus, $V \neq u \cdots u\bar{u}$ and $W \neq \bar{t}t \cdots t, u, t = 0$ or 1, and $SP(V, W)$ degrades from an *LRL-path* to an *R-path* or an *L-path*. If $V = uu \cdots u$ and $W = \overline{uu} \cdots \bar{u}$, where $u = 0$ or 1, then the node-disjoint paths can be constructed easily. Also, if only one of them is $uu \cdots u$, then this is a special case in Case 2. A simple way to construct $SP(V, W)$ is to construct an *R-path*(1). Another path is obtained by shifting in a string ϕ before W. Then, the two node-disjoint paths can be viewed as two strings $w_m w_{m-1} \cdots w_1 v_m v_{m-1} \cdots v_1$ and $w_m w_{m-1} \cdots w_1 \phi v_m v_{m-1} \cdots v_1$ without any common substring of length m except V and W themselves.

Lemma 5. *Strings* $w_m w_{m-1} \cdots w_1 v_m v_{m-1} \cdots v_1$ *and* $w_m w_{m-1} \cdots w_1 \phi v_m v_{m-1} \cdots v_1$ *have no common substring of length m except* $w_m w_{m-1} \cdots w_1$ *and* $v_m v_{m-1} \cdots v_1$, *while* $\phi = \overline{v_m w_2 v_{m-1} w_1}$ *and* $V, W \neq 00 \cdots 0$ *or* $11 \cdots 1$.

Proof. It is clear that only five cases are worth discussing as shown in Figure 7. Also, the discussions of these five cases are easy. ∎

In this case, the two node-disjoint paths have lengths m and $m + 4$, respectively.

4.6. Summary

Table 1 gives a summary for our solutions of fault-tolerant routing. Also, it implies that we have an algorithm to find two node-disjoint paths such that one is the shortest path and the other has length at most $m + \log_2 m + 4$. Thus, the following theorem holds:

Theorem 2. *There exist two node-disjoint paths from a source node V to a destination node W in* $G(2, m), V, W \notin \{uu \cdots u, u \cdots u\bar{u}, \bar{u}u \cdots u\}, u = 0$ *or* 1, *such that one is the shortest path and the other path is of length at most* $m + \log_2 m + 4$.

If we do not require that one of the two node-disjoint paths is the shortest, we can always apply the method in Case 4 or Case 5 to find two node-disjoint paths for every other case. Thus, we have another algorithm to find two node-joint paths with lengths at most m and $m + 4$, respectively. The following theorem also holds:

	$\cdots w_5$	w_4	w_3	w_2	w_1	v_m	v_{m-1}	v_{m-2}	v_{m-3}	$v_{m-4} \cdots$
Case 1	$\cdots w_5$	w_4	w_3	w_2	w_1	ϕ_4	ϕ_3	ϕ_2	ϕ_1	$v_m \cdots$
Case 2	$\cdots w_4$	w_3	w_2	w_1	ϕ_4	ϕ_3	ϕ_2	ϕ_1	v_m	$v_{m-1} \cdots$
Case 3	$\cdots w_3$	w_2	w_1	ϕ_4	ϕ_3	ϕ_2	ϕ_1	v_m	v_{m-1}	$v_{m-2} \cdots$
Case 4	$\cdots w_2$	w_1	ϕ_4	ϕ_3	ϕ_2	ϕ_1	v_m	v_{m-1}	v_{m-2}	$v_{m-3} \cdots$
Case 5	$\cdots w_1$	ϕ_4	ϕ_3	ϕ_2	ϕ_1	v_m	v_{m-1}	v_{m-2}	v_{m-3}	$v_{m-4} \cdots$

FIG. 7. Five cases in the proof of Lemma 5.

TABLE 1. A summary of lengths of node-disjoint paths in Cases 1–5.

	Not insert ϕ	Insert ϕ
Case 1	$m - \|X\| + \|V_L\| + \|W_R\|$,	$m + \log_2 m + 3$
		$m - \|X\| + \|V_L\| + \|W_R\|$
	$2(m - \|X\|) + \|V_L\| + \|W_L\|$	$m + \log_2 m + 3$
Case 2	$m - \|X\| + \|V_L\|$,	$m - \|X\| + \|V_L\|$,
	$m - \|X\| + \|V_L\| + 2\|V_R\|$	$m + \log_2 m + 4$
Case 3	$2(m - \|X\|), 2(m - \|X\|)$	
Case 4	$m - \|X\|, 3(m - \|X\|)$	$m - \|X\|, m + 4$
Case 5		$m, m + 4$

Theorem 3. *There exist two node-disjoint paths from a source node V to a destination node W in* $G(2, m), V, W, \notin \{uu \cdots u, u \cdots u\bar{u}, \bar{u}u \cdots u\}, u = 0$ *or* 1, *such that one is R-path*(k) *and the other path is* $W \cdot \phi \cdot V$. *Also, their lengths are* $m - k$ *and* $m + 4$, *where k is the length of the common substring in the left end of V and the right end of W.*

5. CONCLUDING REMARKS

In this paper, we studied the routing problem on the de Bruijn network. A short diameter, bounded degree, and the capability of fault-tolerance compose our motivation to study the de Bruijn graph. The de Bruijn networks are a good family for performing routing. As we have seen, our fault-tolerant routing algorithm finds two node-disjoint paths. One is shortest path and the other has length at most $m + \log_2 m + 4$ in the worst case. In our analysis, there are some exceptional cases when the node is of the form $u \cdots u, \bar{u}u \cdots u$ or $u \cdots u\bar{u}$. We also extend the results of Cases 4 and 5 in Section 4 that two node-disjoint paths of lengths at most m and $m + 4$ can be found, if we do not require that one of them must be the shortest path. In other words, the fault diameter of the undirected $G(2, m)$ is at most $m + 4$.

On the undirected $G(k, m), k > 2$, our shortest path routing algorithm can also be applied to find the shortest path. However, for the node-disjoint path problem, our algorithms cannot be applied since some nodes may replicate in multiple paths. We are now trying to solve the fault-tolerant routing problem on $G(k, m), k > 2$, by using the concept of this paper, that is, one of the paths is the shortest and the others may be longer.

REFERENCES

[1] M. Beale and S.M.S. Lau, Complexity and auto-correlation properties of a class of de Bruijn sequence, Elect Lett 22 (1986), 1046–1047.

[2] J.C. Bermond and P. Fraigniaud, Broadcasting and gossiping in de Bruijn networks, SIAM J Comput 23 (1994), 212–225.

[3] J. Bruck, R. Cypher, and C.T. Ho, Fault–fault de Bruijn and shuffle-exchange networks, IEEE Trans Parallel Distrib Syst 5 (1994), 548–553.
[4] D. Du and F.K. Hwang, Generalized de Bruijn digraphs, Networks 18 (1988), 27–38.
[5] A.H. Esfahanian and S.L. Hakimi, Fault-tolerant routing in de Bruijn communication networks, IEEE Trans Comput C-34 (1985), 777–788.
[6] S.C. Hu and C.B. Yang, Fault tolerance on star graphs, Int J Found Comput Sci 8 (1997), 127–142.
[7] G. Mayhew and S.W. Golomb, Linear spans of modified de Bruijn sequences, IEEE Trans Infor Theory 36 (1990), 1166–1167.
[8] D.K. Pradhan and S.M. Reddy, A fault-tolerant communication architecture for distributed systems, IEEE Trans Comput 31 (1982), 863–870.
[9] M.R. Samatham and D.K. Pradhan, The de Bruijn multiprocessor network: A versatile parallel processing and sorting network for vlsi, IEEE Trans Comput 38 (1989), 567–581.
[10] M.A. Sridhar, The undirected de Bruijn graph: Fault tolerance and routing algorithms, IEEE Trans Circuits Syst-I: Fundam Theory Appl 39 (1992), 45–48.
[11] M.A. Sridhar and C.S. Raghavendra, Fault-tolerant networks based on the de Bruijn graph, IEEE Trans Comput 40 (1991), 1167–1174.