Joint Conference of 1996 International Computer Symposium
December 19~21, Kaohsiung, Taiwan, R.O.C.

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

Jyh-Wen Mao and Chang-Biau Yang
Department of Applied Mathematics
National Sun Yat-sen University
Kaohsiung, Taiwan 80424, Republic of China
{maojw,cbyang}@math.nsysu.edu.tw

Abstract

We first propose a shortest path routing algorithm on a binary de Bruijn network while researchers have not got a shortest one before. The time required for finding the shortest path in a 2^m-node binary de Bruijn network is $O(m^2)$. Then, based on our shortest path routing algorithm, we propose a fault-tolerant routing scheme. It is assumed that at most one node fails in the network. In our scheme, two node-disjoint paths are found. One is the shortest path, and the other path is of length at most $m + \log_2 m + 4$. If the shortest path is not required in the fault-tolerant routing, we can find two node-disjoint paths with lengths m and $m + 4$.

Key words: interconnection network, de Bruijn graph, routing, fault-tolerant.

1 Introduction

In computer networks and distributed systems, performance and fault tolerance are important issues [1–9]. 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 diameter in the presence of faults. From technology considerations, the number of links(degree) of a processor has to be limited, or even better, to keep constant.

The *de Bruijn* graph has got a good deal of attention by researchers as a graph model for networks [1–6, 8–10]. The node set of binary de Bruijn graph

*This research work was partially supported by the National Center for High-Performance Computing of the Republic of China under contract NCHC-86-08-010.

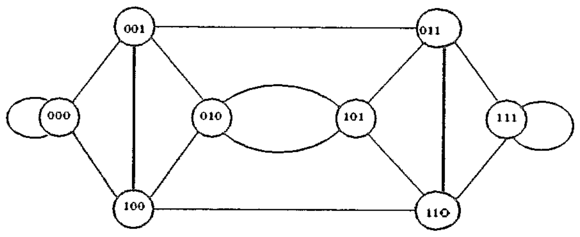

Figure 1: The 8-node binary de Bruijn graph $G(2,3)$.

$G(2,m)$ 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* [9], has maximum degree 4, minimum degree 2, diameter m, and admits a simple routing algorithm. For example, Figure 1 shows the 8-node binary de Bruijn graph. A major goal in topology design is to be able to tolerate failures with relatively small performance degradation. de Bruijn graphs are attractive due to simplicity of 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 low cost for data routing and bounded degree causes 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, 7]. However, this method can not always find the shortest path in an undirected $G(2,m)$. In previous results of fault-tolerant routing on de Bruijn graphs, most researchers focused on the reduction of the increase of fault diameter. Esfahanian and Hakimi [5] have shown that the diameter of $G(k,m)$ increases by at most $\log_k m + 4$ in the presence of up to $2k-2$ faults. Sridhar [9] tightened the bounds of Esfahanian and Hakimi's result [5] significantly. They showed the increase of 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, i.e. $\alpha = (1+\sqrt{5})/2$. 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.

Because of the progress of VLSI technology, the computer components seldom misbehave. It is not a good idea to transmit data between two nodes via two or more node-disjoint paths with same length more than m in $G(k,m)$. In this paper, we shall study the routing problem on undirected binary de Bruijn network, $G(2,m)$. We shall first propose a shortest path routing algorithm, which requires $O(m^2)$ time. Such an algorithm has never been proposed in the previous result. Then, we also 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 still uses the shortest path to transmit data between two nodes if no node fails on the path. If the shortest path is not required in the fault-tolerant routing, we can find two node-disjoint paths with length m and $m+4$.

The rest of this paper is organized as follows. Section 2 gives some notations. The shortest path routing algorithm and fault-tolerant routing algorithm are given in Sections 3 and 4, respectively. And finally, some conclusions are 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 string X is denoted as X_{left} and X_{right}, respectively. We will use *L-operation* and *R-operation* 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. 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, in this paper, a string or a substring may be empty.

3 The Shortest Path Routing Algorithm

The diameter of binary de Bruijn graph $G(m,2)$ has been proved to be m by Parhan and Reddy [7]. 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 *L-path(m)* since it begins at w_m and takes m *L-operations.* We can also establish *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, i.e. *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 [7] proposed a routing method based on the comparison of the lengths of *L-string* and *R-string.* If the length of L-string is greater than that of *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 counter example 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 the source node is $V = 1011011$ and the destination node is $W = 1011010$. Thus, *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 $1011011 \rightarrow 00101101 \rightarrow 1011010$ is the shortest path from V to W, and it takes only two steps. With this example, we conclude that it is impossible to find the shortest path for each pair of nodes only by the consideration of *L-string* and

R-string.

Let V and W be the source and destination nodes, respectively. Suppose 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 performs $|W_R|$ *L-operations* to correct $|W_R|$ bits on the right side of X in W, then performs $|W_L|$ *L-operations* and finally, performs $|W_L|$ *R-operations* to finish this routing process. Let A, B, C and D are some arbitrary strings of proper lengths and the path of 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* and 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 $m - |X| + min(|V_L| + |W_R|, |V_R| + |W_L|)$. For a starting position in V and a starting position in W, a common string X can be found by comparing every bit of V and the corresponding bit of W. The desired X can be obtained by examining every possible starting position in V. This method requires $O(m^2)$ time. We summarize the above shortest routing algorithm as follow.

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 $|V_L|$, $|V_R|$, $|W_L|$, $|W_R|$. Thus, the time complexity is $O(m^2)$, where m is the number of bits to represent a node 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 method can be achieved by shifting the node's binary representation left or right. It is clear that for routing data from V to W, we have to perform some *L-operations* or *R-operations* from V, via some intermediate nodes, to W.

Suppose at least two nonempty 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 can not be shifted out. Then, they can not be corrected to the bits in W, and we can not 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. And, the minimal number of steps of the substitution of W_L for V_L is to move 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. And, 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 don't discuss of 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 the shortest path $SP(V,W)$ is an *LRL-path*, i.e. $V_{L_{right}} \neq W_{L_{right}}$, $V_{R_{left}} \neq W_{R_{left}}$, $m - |X| + |V_L| + |W_R| \leq m$ 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 which the numbers are possible matching positions between two strings. In Figure 3, Cases (a) through (d) illustrate the comparison of one node in P_1 and one node in P_3. And, 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.

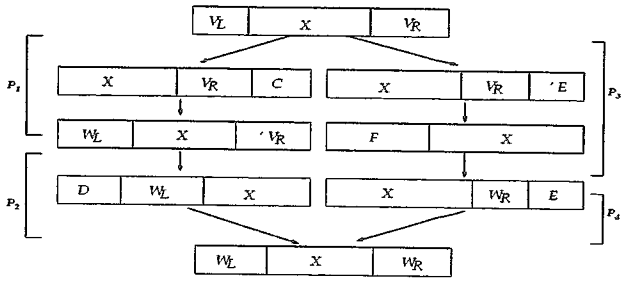

Figure 2: Two node-disjoint paths in Case 1.

Case (a). It implies every bit of X is equal to $\overline{W_{L_{right}}}$ since $F = (\overline{W_{L_{right}}})^*$. 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 position 2 of Case (b). In position 2-2, $W_L \cdot X$ is a substring of $X \cdot V_R$. Thus, we would 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

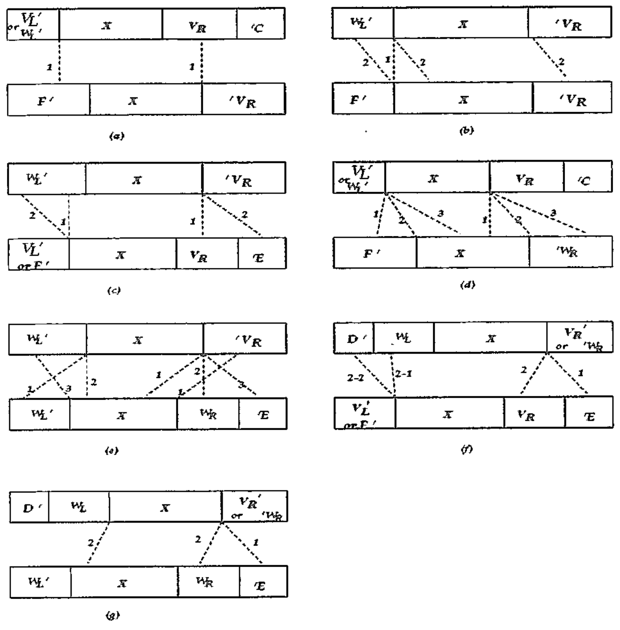

Figure 3: The comparison of two nodes in two paths.

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, of string $W \cdot \phi \cdot V$ 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 $D \cdot W_L \cdot X \cdot V_R \cdot C$ is $|W_R|+ |W_L|$ more than that of string $X \cdot V_R \cdot C$, where $|D| = |W_R|$. We conclude that there are 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 will 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 $D \cdot W_L \cdot X \cdot W_R$. Because we can not make sure that each of $\delta' \cdot' V$ and $W' \cdot' \delta$ with length m is not a

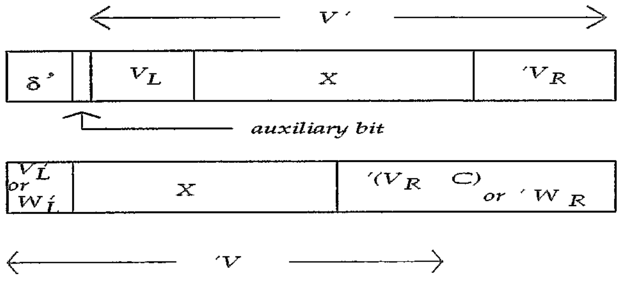

Figure 4: Construction of ϕ based upon δ.

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 can not 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 up to $|\delta| + 2 = log_2 2m + 2 = log_2 m + 3$ 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 little difference from Case 1, i.e., W_R degrades to an empty string. Thus, $SP(V,W)$ degrades to an *RL-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.

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. And, Case (c) and Case (d) illustrate the comparison of one node in P_2 and one

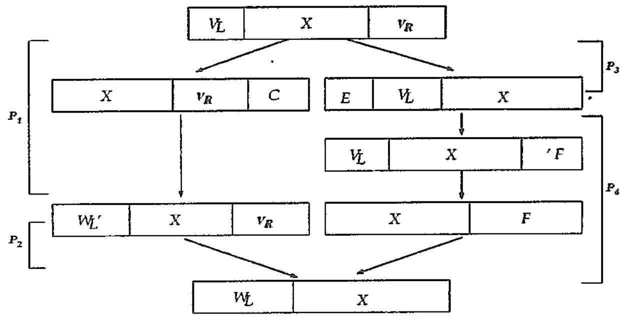

Figure 5: Two node-disjoint paths in Case 2.

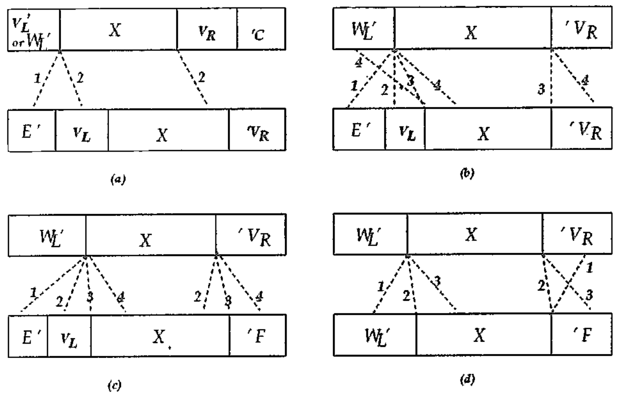

Figure 6: The comparison of two nodes in two paths.

node in P_4, 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 $3m$ while $|X|$ 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 $V \rightarrow \cdots \rightarrow \phi' \cdot' V \rightarrow \cdots \rightarrow W' \cdot \phi \cdot' V \rightarrow \cdots \rightarrow W' \cdot' \phi \rightarrow \cdots \rightarrow W$ and $SP(V, W)$ have no common intermediate node. The method to construct ϕ is similar to that of Case 1. Since $|W_R| = 0$, ϕ_{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) $W = \overline{u}u \cdots u$ or $uu \cdots u$. For the first case, it implies $V_{L_{right}} = \overline{t}$, $W_{L_{right}} = t$ and $V_{R_{left}} = \overline{t}$. Thus, ϕ can be set to $t\overline{t}\overline{V_{R_{left-1}}}t$. 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 $\overline{u}u \cdots u$ or $uu \cdots u$. $SP(V, W)$ would be $V = V_L \cdot X \cdot V_R \rightarrow \cdots \rightarrow X \cdot V_R \cdot \overline{u} \cdots \overline{u} \rightarrow \cdots \rightarrow W_L' \cdot X \cdot V_R \rightarrow \cdots \rightarrow W_L \cdot X = W$ and another path would be $V = V_L \cdot X \cdot V_R \rightarrow \cdots \rightarrow X \cdot V_R \cdot u \cdots u \rightarrow \cdots \rightarrow V_R \cdot u \cdots u \rightarrow \cdots \rightarrow W$, which is an *L-path* and whose length is less than or equal to m. Since $V_{R_{left}} = \overline{u}$ and $X = uu \cdots u$ in this case, it is easy to show that the above two paths are node-disjoint while $|V_R| \leq \frac{m}{2}$. And, it is not hard to show that the argument also holds while $|V_R| > \frac{m}{2}$.

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 $V = V_L \cdot X \rightarrow \cdots \rightarrow X \cdot C \rightarrow \cdots \rightarrow W_L \cdot X = W$ and $V = V_L \cdot X \rightarrow \cdots \rightarrow X \cdot \overline{C} \rightarrow \cdots \rightarrow W_L \cdot X = W$, 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. And, the shortest path from V to W can be represented as $X \cdot V_R \rightarrow \cdots \rightarrow W_L' \cdot X \cdot' V_R \rightarrow \cdots \rightarrow W_L \cdot X$, i.e. $SP(V, W)$ 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 routing path from V to W that is node-disjoint to $SP(V, W)$.

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

Proof: Let every bit of 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 routing from V to W. Another solution is similar to Sridhar's method [9], 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 $\phi' \cdot X \cdot' V_R$. And, 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 : $\phi$.

  $k = |X|$, $i = \lfloor \frac{k}{2} \rfloor$.
  $d = 0$ 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 $(b_1 = \overline{u})$ and $(c_q = \overline{u})$ then $\phi = u$.
      if $(c_q = u)$ and $(b_1 = \overline{u})$ then $\phi = \overline{c_q}\,\overline{b_2}\,\overline{u}\,\overline{b_1}$.
      if $(b_1 = u)$ and $(c_q = \overline{u})$ then $\phi = \overline{c_q}\,\overline{u}\,\overline{c_{q-1}}\,\overline{b_1}$.
  else
      if $X$ has a period of length $i-j$, $-1 \leq j \leq i-1$,
      then
          if $x_{i-j} = c_q$ then $\phi = \overline{c_q}\,\overline{x_{i+j+f+2}}\,\overline{b_1}$
          else if $x_{i+j+f+1} = b_1$ then $\phi = \overline{c_q}\,\overline{x_{i-j-1}}\,\overline{b_1}$.
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 omitted here due to the space limitation. ∎

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 $V = u\cdots u\overline{u}$ or $V = u\cdots uu$ if only one of them is an exception node. For $V = u\cdots u\overline{u}$, we can solve this problem by using a simple trick on V. Let $P = u\cdots u\overline{uu}$ and $\phi = \overline{uw_{l_2}}\overline{u}u$. Then the routing path from P to W is following the string $W \cdot \phi \cdot P$ $(= W_L \cdot u\cdots u \cdot \phi \cdot u\cdots u\overline{uu})$. 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 $u\cdots u\overline{u}$ and $\overline{u}u\cdots u$. 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 $|V_L| = 0$ and $|W_R| = 0$. Thus, $V \neq u\cdots u\overline{u}$ and $W \neq \overline{t}t\cdots t$, $u, t = 0$ 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 $W = \overline{uu}\cdots\overline{u}$, where $u = 0$ or 1, then the node-disjoint paths can be constructed easily. And, 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. And, 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. And, it implies Theorem 2 holds.

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

	$\cdots w_5$	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}	$v_{m-4}\cdots$

Figure 7: Five cases in the proof of Lemma 5.

Table 1: A summary of lengths of node-disjoint paths in Cases 1 through 5.

	not insert ϕ	insert ϕ
Case 1	$m-\lvert X\rvert+\lvert V_L\rvert+\lvert W_R\rvert$, $2(m-\lvert X\rvert)+\lvert V_L\rvert+\lvert W_L\rvert$	$m-\lvert X\rvert+\lvert V_L\rvert+\lvert W_R\rvert$, $m+log_2m+3$
Case 2	$m-\lvert X\rvert+\lvert V_L\rvert$, $m-\lvert X\rvert+\lvert V_L\rvert+2\lvert V_R\rvert$	$m-\lvert X\rvert+\lvert V_L\rvert$, $m+log_2m+4$
Case 3	$2(m-\lvert X\rvert)$, $2(m-\lvert X\rvert)$	
Case 4	$m-\lvert X\rvert$, $3(m-\lvert X\rvert)$	$m-\lvert X\rvert$, $m+4$
Case 5		m, $m+4$

5 Concluding Remarks

In this paper, we study the routing problem on the de Bruijn network. 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 vertex-disjoint paths. One is shortest path and the other has length at most $m + log_2m + 4$ in the worst case. In our analysis, there are some exceptional cases when the node is of the form $u\cdots u$, $\overline{u}u\cdots u$ or $u\cdots u\overline{u}$. By cases 4 and 5 in Section 4, we also conclude that the increase of fault diameter is at most 4 if we do not require that one of them must be the shortest path.

References

[1] M. Beale and S. M. S. Lau, "Complexity and auto-correlation properties of a class of de bruijn sequence," *Electronic Letters*, Vol. 22, No. 20, pp. 1046–1047, May 1986.

[2] J. C. Bermond and P. Fraigniaud, "Broadcasting and gossiping in de bruijn networks," *SIAM Journal on Computing*, Vol. 23, No. 1, pp. 212–225, Feb. 1994.

[3] J. Bruck, R. Cypher, and C. T. Ho, "Fault-fault de bruijn and shuffle-exchange networks," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 5, No. 5, pp. 548–553, May 1994.

[4] D. Du and F. K. Hwang, "Generalized de bruijn digraphs," *Networks*, Vol. 18, No. 1, pp. 27–38, 1988.

[5] A. H. Esfahanian and S. L. Hakimi, "Fault-tolerant routing in debruijn communication networks," *IEEE Transactions on Computers*, Vol. C-34, No. 9, pp. 777–788, Sep. 1985.

[6] G. Mayhew and S. W. Golomb, "Linear spans of modified de bruijn sequences," *IEEE Transactions on Information Theory*, Vol. 36, No. 5, pp. 1166–1167, 1990.

[7] D. K. Pradhan and S. M. Reddy, "A fault-tolerant communication architecture for distributed systems," *IEEE Transaction on Computers*, Vol. 31, No. 9, pp. 863–870, Sep. 1982.

[8] M. R. Samatham and D. K. Pradhan, "The de bruijn multiprocessor network: A versatile parallel processing and sorting network for vlsi," *IEEE Transactions on Computers*, Vol. 38, No. 4, pp. 567–581, 1989.

[9] M. A. Sridhar, "The undirected de bruijn graph: Fault tolerance and routing algorithms," *IEEE Transactions on Circuits and Systems-I: Fundamental Theory and Applications*, Vol. 39, No. 1, pp. 45–48, 1992.

[10] M. A. Sridhar and C. S. Raghavendra, "Fault-tolerant networks based on the de bruijn graph," *IEEE Transactions on Computers*, Vol. 40, No. 10, pp. 1167–1174, Oct. 1991.