Fault Free Shortest Path Routing on the de Bruijn Networks*

Ngoc Chi Nguyen, Nhat Minh Dinh Vo and Sungyoung Lee

Computer Engineering Department, Kyung Hee Univerity
1, Seocheon, Giheung, Yongin, Gyeonggi 449-701 KOREA
{ncngoc, vdmnhat, sylee}@oslab.khu.ac.kr

Abstract. It is shown that the de Bruijn graph (dBG) can be used as an architecture for interconnection networks and a suitable structure for parallel computation. Recent works have classified dBG based routing algorithms into shortest path routing and fault tolerant routing but investigation into shortest path in failure mode in dBG has been nonexistent. In addition, as the size of the network increase, more faults are to be expected and therefore shortest path algorithms in fault free mode may not be suitable routing algorithms for real interconnection networks, which contain several failures. Furthermore, long fault free path may lead to high traffic, high delay time and low throughput.In this paper we investigate routing algorithms in the condition of existing failure, based on the Bidirectional de Bruijn graph (BdBG). Two Fault Free Shortest Path (FFSP) routing algorithms are proposed. Then, the performances of the two algorithms are analyzed in terms of mean path lengths. Our study shows that the proposed algorithms can be one of the candidates for routing in real interconnection networks based on dBG.

1 Introduction

For routing in dBG, Z. Liu and T.Y. Sung [1] proposed eight cases shortest paths in BdBG. Nevertheless, Z. Liu's algorithms do not support fault tolerance. J.W. Mao [4] has also proposed the general cases for shortest path in BdBG (case RLR or LRL). For fault tolerance issue, he provides another node-disjoint path of length at most $k + log_2 k + 4$ (in dBG(2,k)) beside shortest path. However, his algorithm can tolerate only one failure node in binary de Bruijn networks and it cannot achieve shortest path if there is failure node on the path.

Considering limitations of routing in dBG, we intend to investigate shortest path routing in the condition of failure existence. Two Fault Free Shortest Path (FFSP) routing algorithms are proposed. Time complexity of FFSP2 in the worst case is $0(2^{\frac{k}{2}+1}d)$ in comparison with $0((2d)^{\frac{k}{2}+1})$ of FFSP1 (in dBG(d,k) and k=2h). Therefore, FFSP2 is our goal in designing routing algorithm for large network with high degree.

* This research work has been partially supported by Korea Ministry of Information and Communications' ITRC Program joint with Information and Communication University

The rest of this paper is organized as follows. Background is discussed in section 2. In section 3, FFSP routing algorithms are presented. Performance analysis for FFSP routing algorithms is carried in section 4. Finally, some conclusions will be given in Section 5.

2 Background

The BdBG graph denoted as BdBG(d,k)[1] has N=d^k nodes with diameter k and degree 2d. If we represent a node by $d_0d_1...d_{k-2}d_{k-1}$, where $d_j \in 0, 1, ..., (d-1)$, 0≤j≤(k-1), then its neighbor are represented by $d_1...d_{k-2}d_{k-1}p$(L neighbors, by shifting left or L path) and $pd_0d_1...d_{k-2}$(R neighbors, by shifting right or R path), where $p = 0, 1, ..., (d-1)$. We write if the path $P = R_1L_1R_2L_2$ consists of an R-path called R_1, followed by an L-path called L_1, an R-path called R_2, an L-path called L_2, and so on, where subscripts are used to distinguish different sub-paths. Subscripts of these sub-paths can be omitted if no ambiguity will occur, e.g., $P = R_1LR_2$ or P=RL.

The following fig. 1a shows us an example for BdBG(2,4). Fig. 1b shows us eight cases of shortest path routing on BdBG. The gray areas are the maximum substring between source (s) and destination (d). The number inside each block represents the number of bits in the block.

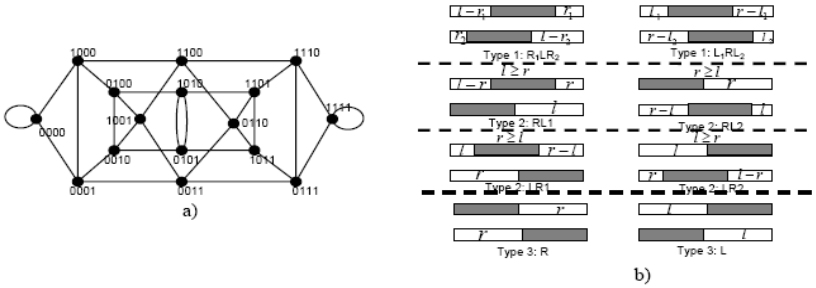

Fig. 1. a)The BdBG(2,4); b)Shortest path types[1].

3 Fault Free Shortest Path Routing Algorithms

By examining this example, finding shortest path between S 1110121100 and D 0012111001 (dBG(3,10)), we can easily see that methods provided by Liu and Mao [1][4] cannot provide FFSP. Their methods determine the maximum matched string, calculate path length corresponding with the matched string and then shifting are performed to finish routing process. In this case, the matched string is "00", path length is 8 and shortest path is 1110121100 → 1101211001

→ 1012110012 → 0121100121 → 1211001211 → 2110012111 → 1100121110 → 1001211100 → 0012111001 (A). If node 1012110012 is failed, then Liu's algorithm is failed in finding shortest path. Mao's algorithm can only work in binary dBG, hence it fails in dBG(3,10).

In order to find shortest path in the condition of failure existing, we cannot base on string matching concept. A shortest path found by string matching cannot be used if there is a fault in the path (as shown in the previous example). Therefore, a new concept of multi level discrete set is proposed (definition 1). By using multi level discrete set, several paths of a specific source destination pair are provided (redundancy is provided). In the above example, we can provide 3 shortest paths from S to D. Those are A; 1110121100 → 1111012110 → 1111101211 → 1111012111 → 1110121110 → 1101211100 → 1012111001 → 0121110011 → 0012111001 (B); and 1110121100 → 2111012110 → 1211101211 → 0121110121 → 0012111012 → 0001211101 → 0000121110 → 0001211100 → 0012111001 (C). In the case A is a failure path, we can use other 2 FFSP B and C. For building our algorithms, we assume that there is a separately protocol which detects failure nodes and then let other nodes know in periodically. Note that, level is defined simply in terms of distance from the root (level 1).

This section is organized as follows, from definition 1 to FFSP1, we state some concepts how to provide several shortest paths of a specific source and destination, and how to find FFSP among these shortest paths. Through definition 2 to FFSP2, we state how to improve the performance of our algorithm.

Definition 1: *the level m^{th} discrete set (DS_m) is a set which contains all neighbors of each element in discrete set level m-1; in the constraint that there is no existent element of discrete set level m coincides with another element of discrete set level q^{th} ($q \leq m$) or failure node set.*

Lemma 1: *DS_m is fault free.*

Lemma 2: *all the neighbors of a node belong to DS_m are in DS_{m-1}, DS_m and DS_{m+1}, except failure nodes.*

Proof: obviously we see that DS_1 and DS_2 contain all the neighbors of DS_1 except failure nodes; DS_1, DS_2 and DS_3 contain all the neighbors of DS_2 except failure nodes. So Lemma 2 is right at m=1,2. Assuming that lemma 2 is right until p, now we prove it is right at p+1. Suppose it's wrong at p+1. That means there exist a neighbor A of an element B$\in DS_{p+1}$, and A $\in DS_i, i < p$. Because lemma 2 is right until p, hence all the neighbors of A are in DS_{i-1}, DS_i and DS_{i+1} except failure nodes. Therefore, there exists an element B'$\in DS_{i-1}, DS_i$ or DS_{i+1}, and B'=B. It contradicts with definition 1. So Lemma 2 is right at p+1. Following inductive method, lemma 2 is proved.

Lemma 3: *there exists no neighbor of any element of DS_m, which is a duplicate of any element of DS_h, $\forall h \leq m\text{-}2$.*

Proof: suppose there is a neighbor A of an element B $\in DS_m$ duplicates with an element A' of DS_h (h $\leq$ m-2). Following Lemma 2, all the neighbors of A' are in DS_{h-1}, DS_h and DS_{h+1}. Therefore, there must exist a neighbor B' of A' in level h-1 or h or h+1, and B'=B. It contradicts with definition 1.

Corollary 1: *for duplicate checking at the next level of DS_q, it is not necessary to check with any element of DS_m,$\forall m \leq q\text{-}2$.*

By assigning source node S to DS_1, then expanding to the higher level, we have the following theorem.

Theorem 1: *in BdBG(d,k), we can always find a FFSP from node $S \in DS_1$ to node $A_x \in DS_x (\forall x \leq k)$, if it exists.*

Proof: we use inductive method to prove this theorem. When x=1, 2, theorem 1 is right. Assuming that theorem 1 is right until m, m $\leq$ k. Now we prove it is right until m+1. Suppose that path from S to A_{m+1} is not the FFSP. Then we have the following cases,

• There exist $A_p \in DS_p$, $A_p = A_{m+1}$and $p < m+1$.It contradicts definition 1.

• There exists a FFSP, S$\rightarrow B_1 \rightarrow B_2 \rightarrow ... \rightarrow B_k \rightarrow ... \rightarrow B_z \rightarrow ... \rightarrow A_{m+1}$, and B_k, B_{k+1},..., B_z not belonging to any DS_i ($\forall$ i$\leq$m+1). Because $B_{k-1} \in DS_j$ (j$\leq$m+1). Following Lemma 2, all the neighbors of B_{k-1} are in $DS_{j-1} or DS_j or DS_{j+1}$, except failure nodes. Therefore, B_k must be a failure node.

$\rightarrow$Theorem 1 is right at m+1. Theorem 1 is proved.

Corollary 2: *path length of a path from $S \in DS_1$ to $A_x \in DS_x$ is x-1.*

Fault free shortest path algorithm 1 (FFSP1) is proposed as a result of theorem 1 (shown in fig. 2a). It can always find FFSP in all cases (fault free mode, arbitrary failure mode) if the network still remain connected.

Proof of FFSP1: suppose path s$\rightarrow ... \rightarrow a_{ip} \rightarrow b_{jk} \rightarrow ...\rightarrow$d is not FFSP, and then we have the following cases,

• There exist a FFSP s$\rightarrow ... \rightarrow a_{i'p'} \rightarrow b_{j'k'} \rightarrow ...\rightarrow$d (i'$\leq$i, j'$\leq$j). It contradicts with the above assumption that a_{ip} and b_{jk} are the first neighbors between discrete sets A and B.

• There exist a FFSP s$\rightarrow ... \rightarrow a_{i'p'} \rightarrow c_1 \rightarrow ... \rightarrow c_m \rightarrow b_{j'k'} \rightarrow ...\rightarrow$d ($i' < i, j' < j$), and $c_1, c_2, ..., c_m$ do not belong to any discrete set A_p or B_q (p$\leq$i, q$\leq$j). Due to $a_{i'p'} \in A_{i'}$ and following lemma 2, all the neighbors of $a_{i'p'}$ are in $A_{i'-1}, A_{i'}$ and $A_{i'+1}$ except failure nodes. Therefore c_1 must be a failure node.

Example 1: we want to find a FFSP from source 10000 to destination 01021, failure node 00102 (dBG(3,5)).

Applying FFSP1, we have, A_1 = (10000) B_1 = (01021) A_2 = (00000, 00001, 00002, 01000, 11000, 21000) B_2 = (10210, 10211, 10212, 00102, 10102, 20102).

However, 00102 is a failure node. So B_2=(10210, 10211, 10212, 10102, 20102).

A_3 = (20000, 00010, 00011, 00012, 00020, 00021, 00022, 10001, 10002, 00100, 10100, 20100, 01100, 11100, 21100, 02100, 12100, 22100)

Then we find that 02100 and 10210 in A_3 and B_2 are the first neighbors. FFSP is found by tracking back from 02100 to 10000 and 10210 to 01021. We have FFSP 10000 $\rightarrow$ 21000 $\rightarrow$ 02100 $\rightarrow$ 10210 $\rightarrow$ 01021. In this example, FFSP1 can provide 2 shortest paths (in the case of no failure node) 10000 $\rightarrow$ 21000 $\rightarrow$ 02100 $\rightarrow$ 10210 $\rightarrow$ 01021 and 10000 $\rightarrow$ 00001 $\rightarrow$ 00010 $\rightarrow$ 00102 $\rightarrow$ 01021. We pick up one FFSP 10000$\rightarrow$21000$\rightarrow$02100$\rightarrow$10210$\rightarrow$01021 (node 00102 is fail).

Furthermore, we shall see that other elements like 00000, 00002, 01000, 11000 in A_2 are useless in constructing a FFSP. So, eliminating these elements can reduce the size of A_3 (reduce the cost at extending to next level) and improve

the performance of our algorithm. It shows the motivation of FFSP2. Before investigating FFSP2, we give some definition and theorem.

Definition 2: *a dominant element is an element which makes a shorter path from source to a specific destination, if the path goes through it.*

Example 2: from the above example 1 we have 2 shortest paths (in the case 00102 is not a failure node) 10000 → 21000 → 02100 → 10210 → 01021 and 10000 → 00001 → 00010 → 00102 → 01021. Thus 00001 and 21000 are dominant elements of A_2, because they make shorter path than others of A_2.

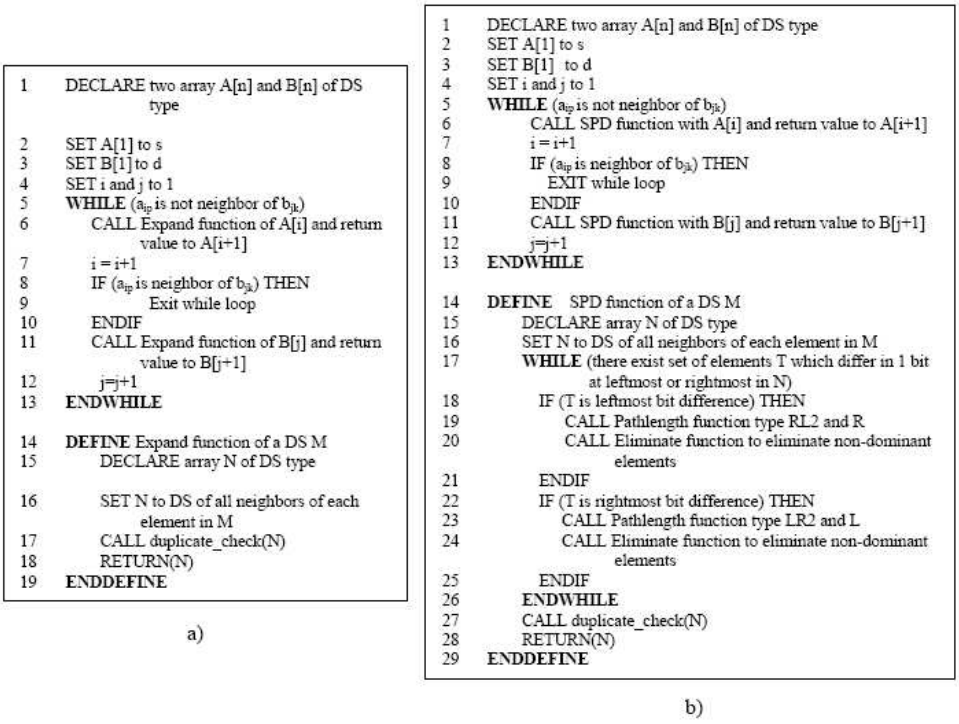

Fig. 2. a)Fault Free Shortest Path Algorithm 1 (FFSP1); b)Fault Free Shortest Path Algorithm 2 (FFSP2).

Therefore, by eliminating some non-dominant elements in a level, we can reduce the size of each level in FFSP1 and hence, improve the performance of FFSP1. A question raised here is how we can determine some dominant elements in a DS_k and how many dominant elements, in a level, are enough to find FFSP. The following theorem 2 is for determining dominant elements and corollary 3 answer the question, how many dominant elements are enough.

Theorem 2: *If there are some elements different in 1 bit address at leftmost or rightmost, the dominant element among them is an element which has shorter*

path length toward destination for cases RL2, R (shown in fig. 1b) for leftmost bit difference and LR2, L for rightmost bit difference.

Proof: as showing in fig. 1b, there are eight cases for shortest path. Only four cases RL2, R, LR2 and L make different paths when sources are different in leftmost bit or rightmost bit.

Example 3: following example 1, we check the dominant characteristic of three nodes A 01000, B 11000 and C 21000 (in A_2) to destination D 01021. Three nodes A, B and C are leftmost bit difference. So, type RL2, R are applied.

• Apply type R: the maximum match string between A 01000 and D 01021 is 0, between B **1**1000 and D 0102**1** is 1, and between C **21**000 and D 010**21** is 2 → min path length is 3, in case of node C.

• Apply type RL2: the maximum match string [5] between A **0**1000 and D 01**0**21 is 1 (path length: 6), between B **1**1000 and D 0**1**021 is 1 (path length: 7), and between C **21**000 and D 010**21** is 2 (same as case R) → min is 3, node C.

Therefore, minimum path length is 3 and dominant element is C.

Corollary 3: *when we apply theorem 2 to determine dominant elements, the maximum elements of DS_{m+1} are 2p(p is the total elements of DS_m).*

Proof: the maximum elements of DS_{m+1} by definition 1 are 2pd (dBG(d,k)). We see that in 2pd there are 2p series of d elements which are different in 1 bit at leftmost or rightmost. By applying theorem 2 to DS_{m+1}, we obtain 1 dominant element in d elements differed in 1 bit at leftmost or rightmost.

Fault Free Shortest Path Algorithm 2 (FFSP2) is proposed in fig. 2b.

The condition in line 5 and line 8 (fig. 2a, 2b) let us know whether there exists a neighbor of array A and B of discrete set, $\forall a_{ip} \in$A[i],$\forall b_{jk} \in$B[j] . The SPD(M) function, line 14 fig. 2b, finds the next level of DS M (DS N) and eliminates non-dominant elements in N followed theorem 2. Expand(M) function, line 14 fig. 2a, finds the next level of DS M. Pathlength type p function, line 19,23 fig. 2b, checks path length followed type p of each element in T toward destination. Eliminate function, line 20, 24, eliminates element in T, which has longer path length than the other. The duplicate_check(N) function, line 17 fig. 2a and line 27 fig. 2b, check if there is a duplicate of any element in N with other higher level DS of N. For duplication checking, we use the result from corollary 1. Then, we get FFSP by going back from a_{ip} to s and b_{jk} to d.

Example 4: we try to find FFSP as in example 1. By applying FFSP2, we have, A_1 = (10000) B_1 = (01021) A_2 = (00001, 21000) B_2 = (10210, 00102). However, 00102 is a failure node. So B_2 becomes (10210).

A_3 = (00010, 10000, 10001, 02100). However, node 10000 coincides with 10000 of A_1. So A_3 becomes (00010, 10001, 02100). Then we find that 02100 and 10210 in A_3 and B_2 are the first neighbors. FFSP is found by tracking back from 02100 to 10000 and 10210 to 01021. We have FFSP 10000 → 21000 → 02100 → 10210 → 01021.

4 Performance analysis for FFSP1 and FFSP2

Mean path length is the significant to analyze and compare our algorithm to others. Z. Feng and Yang [2] have calculated it based on the original formula, Mean path length = $\frac{Totalinternaltraffic}{Totalexternaltraffic}$ for their routing performance. We can use the above equation to get the mean path length in the case of failure. We assume that failure is random, and our network is uniform. That means the probability to get failure is equal at every node in the network.

Table 1 shows the results in the simulation of mean path length using six algorithms, SCP[3], RFR, NSC, PMC[2], FFSP1 and FFSP2. Our two algorithms show to be outstanding in comparison with the four algorithms. They always achieve shorter mean path length than the other algorithms.

Table 1. Mean path length of FFSP1, FFSP2 in comparison with others.

d,k	no. of nodes	Mean path lengths								
		SCP (fault free mode)	RFR (fault free mode)	NSC (fault free mode)	PMC (fault free mode)	FFSP1/2 (fault free mode)	FFSP1 (1 node fail)	FFSP2 (1 node fail)	FFSP1 (2 nodes fail)	FFSP2 (2 nodes fail)
2,2	4	1.167	1.167	1.167	1.167	1.167	1.333	1.333	1	1
2,3	8	1.643	1.643	1.643	1.643	1.643	1.762	1.762	1.733	1.733
2,4	16	2.258	2.188	2.146	2.142	2.142	2.1	2.21	2.286	2.286
2,5	32	2.984	2.796	2.794	2.766	2.754	2.787	2.794	2.285	2.832
2,6	64	3.801	3.653	3.551	3.495	3.453	3.467	3.483	3.487	3.506
3,2	9	1.417	1.471	1.417	1.417	1.417	1.429	1.429	1.476	1.476
3,3	27	2.128	2.105	2.007	2.077	2.077	2.095	2.095	2.117	2.117
3,4	81	2.978	2.911	2.865	2.849	2.833	2.84	2.842	2.846	2.848
3,5	243	3.907	3.800	3.755	3.736	3.674	3.676	3.696	3.678	3.698
3,6	729	4.875	4.775	4.704	4.722	4.572	4.573	4.626	4.573	4.627
4,2	16	1.550	1.550	1.550	1.550	1.550	1.552	1.552	1.56	1.56
4,3	64	2.369	2.343	2.321	2.321	2.321	2.327	2.327	2.335	2.335
4,4	256	3.298	3.251	3.214	3.218	3.178	3.18	3.185	3.184	3.189
4,5	1024	4.273	4.207	4.172	4.209	4.1	4.101	4.13	4.101	4.13
5,2	25	1.633	1.633	1.633	1.633	1.633	1.634	1.634	1.636	1.636
5,3	125	2.510	2.491	2.471	2.471	2.471	2.473	2.473	2.476	2.476
5,4	625	3.471	3.438	3.41	3.437	3.378	3.379	3.385	3.379	3.385
6,2	36	1.690	1.690	1.690	1.690	1.690	1.691	1.691	1.693	1.693
6,3	216	2.601	2.585	2.570	2.570	2.570	2.571	2.571	2.573	2.573

This section is completed with study in time complexity of our algorithms. As A. Sengupta [9] has shown that dBG(d,k) has connectivity of d-1. Hence, our time complexity study is based on assumption that the number of failures is at most d-1 and our study is focused on large network with high degree (d>>1). Therefore, diameter of our network in this case is k. We have the following cases,

• For FFSP1, the second level DS lies in the complexity class 0(2d) , the third level DS lies in the complexity class 0(2d(2d-1))≈ $0(4d^2)$, the fourth lies in $0(2d(2d-1)^2) \approx 0(8d^3)$, etc... Hence, time complexity of FFSP1 lies in the complexity class $0((2d)^n)$, the value of n equals to the maximum level DS provided by FFSP1. In the worst case, time complexity of FFSP1 lies in $0((2d)^{\frac{k}{2}+1})$ (k=2h),

or $0((2d)^{\frac{k+1}{2}})$ (k=2h+1), k is maximum path length from source to destination (the diameter).

• The computation time of FFSP2 can be divided into 2 parts. One is performing computation on expanding to next level, checking for duplicate and neighboring checking between DS A[m] and B[q]. This part is like FFSP1, the difference is that each DS here grows following a geometric progression with common quotient 2 and initial term 1 (as shown in corollary 3). The other part is performing computation on finding dominant elements. Hence, the second level DS lies in the complexity class 0(2+2d)≈0(2d), the third level DS lies in the complexity class 0(4+4d)≈0(4d), the fourth lies in 0(8+8d)≈0(8d), etc... Hence time complexity of FFSP2 lies in the complexity class $0(2^n d)$, the value of n equals to the maximum level DS provided by FFSP2. FFSP2 would cost us $0(2^{\frac{k}{2}+1}d)$ (k=2h), or $0(2^{\frac{k+1}{2}})$ (k=2h+1) time in the worst cases, k is maximum path length from source to destination (the diameter).

5 Conclusion

We have proposed new concepts, and routing algorithms in dBG(d,k). Our routing algorithms can provide shortest path in the case of failure existence. Our simulation result shows that FFSP2 is an appropriate candidate for the real networks with high degree and large number of nodes, while FFSP1 is a good choice for high fault tolerant network with low degree and small/medium number of nodes. Therefore, the algorithms can be considered feasible for routing in real interconnection networks.

References

1. Zhen Liu, Ting-Yi Sung, "Routing and Transmitting Problem in de Bruijn Networks" IEEE Trans. on Comp., Vol. 45, Issue 9, Sept. 1996, pp 1056 - 1062.
2. O.W.W. Yang, Z. Feng, "DBG MANs and their routing performance", Comm., IEEE Proc., Vol. 147, Issue 1, Feb. 2000 pp 32 - 40.
3. A.H. Esfahanian and S.L. Hakimi, "Fault-tolerant routing in de Bruijn communication networks", IEEE Trans. Comp. C-34 (1985), 777.788.
4. Jyh-Wen Mao and Chang-Biau Yang, "Shortest path routing and fault tolerant routing on de Bruijn networks", Networks, Vol. 35, Issue 3, Pages 207-215 2000.
5. Alfred V. Aho, Margaret J. Corasick, "Efficient String Matching: An Aid to Bibliographic Search", Comm. of the ACM, Vol. 18 Issue 6, June 1975.