- G_i : Mean communication delay on link i.
- g_i : Incremental communication delay on link i.
- h_i : Difference between the incremental node delay at node i and the incremental communication delay on link i.
- L : Set of links.
- N : Set of nodes.
- Nu_j: Set of neutral nodes in V_j.
- Ra_j : Set of active source nodes in V_j.
- Rd_j : Set of idle source nodes in V_j.
- T : Set of terminal nodes.
- V_j : Set of nodes that are children of node j ($j \in B$).
- x_i : Job flow rate from node i to its parent node.
- $\mathbf{x}$: $[x_i]$.
- α_i : Lagrange multiplier.
- β_i : Job processing rate (load) at node i.
- $\boldsymbol{\beta}$: $[\beta_i]$.
- ϕ_i : External job arrival rate at node i.
- ϕ_i' : Total job flow rate into node i.
- $\boldsymbol{\Phi}$: Total external job arrival rate.

References

[1] D.G. Cantor and M. Gerla, "Optimal routing in a packet-switched computer network," *IEEE Trans. Comput.*, vol. C-23, no. 10, pp. 1062–1069, Oct. 1974.

[2] P.J. Courtois, *Decomposability: Queueing and Computer System Applications*. New York: Academic, 1977.

[3] S.C. Dafermos and F.T. Sparrow, "The traffic assignment problem for a general network," *J. Res. National Bureau Standards-B*, vol. 73B, no. 2, pp. 91–118, 1969.

[4] L. Fratta, M. Gerla, and L. Kleinrock, "The flow deviation method: An approach to store-and-forward communication network design," *Networks*, vol. 3, pp. 97–133, 1973.

[5] M.D. Intriligator, *Mathematical Optimization and Economic Theory*. Englewood Cliffs, NJ: Prentice-Hall, 1971.

[6] H. Kameda and A. Hazeyama, "Individual vs. overall optimization for static load balancing in distributed computer systems," Comput. Sci. Rep., Univ. of Electro-Commun., Japan, 1988.

[7] C. Kim and H. Kameda, "An algorithm for optimal static load balancing in distributed computer systems," *IEEE Trans. Comput.*, vol. 41, pp. 381–384, 1992.

[8] L. Kleinrock, *Queueing Systems: Computer Applications*, Vol. 2. New York: Wiley, 1976.

[9] J.G. Lee, W.G. Vogt, and M.H. Mickle, "Optimal decomposition of large-scale networks," *IEEE Trans. Syst., Man, Cybernetics*, vol. SMC-9, no. 7, pp. 369–375, July 1979.

[10] T. Leventhal, G. Nemhauser, and L. Trotter, "A column generation algorithm for optimal traffic assignment," *Transp. Sci.*, vol. 7, pp. 168–176, 1973.

[11] T.L. Magnanti, "Models and algorithms for predicting urban traffic equilibria," *Transportation Planning Models*. M. Florian, Ed. Amsterdam: Elsevier B.V. (North-Holland), 1984, pp. 153–185.

[12] A.B. Nagurney, "Comparative tests of multimodal traffic equilibrium methods," *Transp. Res.-B*, vol. 18B, no. 6, pp. 469–485, 1984.

[13] M. Schwartz, *Computer Communication Network Design and Analysis*. Englewood Cliffs, NJ: Prentice-Hall, 1977.

[14] A.S. Tanenbaum, *Computer Networks*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1988.

[15] A.N. Tantawi and D. Towsley, "A general model for optimal static load balancing in Star Network configurations," *PERFORMANCE'84*, E. Gelenbe Ed., Amsterdam: North-Holland, 1984, pp. 277–291.

[16] A.N. Tantawi and D. Towsley, "Optimal static load balancing in distributed computer systems," *J. ACM*, vol. 32, no. 2, pp. 445–465, Apr. 1985.

Fault-Tolerant de Bruijn and Shuffle-Exchange Networks

Jehoshua Bruck, Robert Cypher, and Ching-Tien Ho

***Abstract*— This paper addresses the problem of creating a fault-tolerant interconnection network for a parallel computer. Three topologies, namely, the base-2 de Bruijn graph, the base-*m* de Bruijn graph, and the shuffle-exchange, are studied. For each topology an $N + k$ node fault-tolerant graph is defined. These fault-tolerant graphs have the property that given any set of k node faults, the remaining N nodes contain the desired topology as a subgraph. All of the constructions given here are the best known in terms of the degree of the fault-tolerant graph. We also investigate the use of buses to reduce the degrees of the fault-tolerant graphs still further.**

***Index Terms*— de Bruijn networks, fault tolerance, reconfiguration, shuffle-exchange networks, spare nodes**

I. Introduction

Many different topologies have been suggested for distributed-memory parallel computers. One of the most important topologies is the hypercube, which has been shown to be efficient in supporting a wide range of algorithms. In fact, several commercial machines have been based on the hypercube topology. However, the degree of the hypercube grows with the number of processors. Because technological considerations limit the degree of the processor, there is a limit to the number of processors that can be connected in a hypercube topology.

Fortunately, several constant-degree topologies have been proposed as possible point-to-point interconnection networks, which have many of the same properties as the hypercube. Most notable among these are the shuffle-exchange [13], the cube-connected cycles [11], and the de Bruijn [1] topologies. All of these topologies are able to support a wide range of algorithms, including those in the classes Ascend and Descend [11], with only a small constant factor slowdown relative to the hypercube. As a result, they appear to be attractive alternatives to the hypercube for the design of massively parallel machines.

There is, however, a serious difficulty with using these constant-degree topologies for massively parallel machines. As the number of processors in a machine increases, so does the likelihood that one or more of the processors or communication links will be faulty. Faults are an extremely serious concern in these constant-degree networks, because most of the efficient algorithms that have been designed for them utilize all of the processors and all of the communication links. As a result, a single processor or link failure can severely degrade the performance of the parallel machine.

Our approach to fault tolerance is based on a graph model introduced by Hayes [9]. In this model, a parallel computer is viewed as being a graph, where the nodes represent processors and the edges represent communication links. The model starts by selecting a target graph G (such as a shuffle exchange). Then a fault-tolerant graph G' is defined such that given any set of k or fewer faults, G' is guaranteed to contain as a subgraph a nonfaulty copy of the graph G. Of course, the challenge is to create a fault-tolerant graph G' that has the smallest possible amount of redundancy, both in terms of the number of extra nodes and the degree.

Manuscript received January 15, 1992; revised May 28, 1993.

The authors are with the IBM Research Division, Almaden Research Center, San Jose, CA 95120.

IEEE Log Number 9216784.

A number of researchers have applied this graph model of fault tolerance to various target graphs [2]–[6]. In this paper, we focus on the creation of fault-tolerant de Bruijn and (point-to-point) shuffle-exchange graphs. We consider only node faults, but it should be noted that edge faults can be tolerated by viewing a node that is incident to the faulty edge as being faulty. All of our constructions use the minimum number of nodes, so if the target graph G has N nodes and if k node faults must be tolerated, our fault-tolerant graph G' will have exactly $N+k$ nodes. All of our constructions also have degrees that are independent of N, the number of nodes in the target graph.

Several researchers have studied fault-tolerance in de Bruijn networks. Esfahanian and Hakimi [8] examined how many node faults a de Bruijn graph can tolerate without becoming disconnected. Samatham and Pradhan [12] used Hayes's graph model to obtain fault-tolerant de Bruijn graphs. Given a de Bruijn graph G as a target graph and the requirement of tolerating k node faults, they select a larger de Bruijn graph G' as their fault-tolerant graph. When the target graph is a base-2 de Bruijn graph with N nodes, their construction yields a fault-tolerant graph with $N^{\log_2(2k+1)}$ nodes and degree $4k+2$. When the target graph is a base-m de Bruijn graph with N nodes, their construction yields a fault-tolerant graph with $N^{\log_m(mk+1)}$ nodes and degree $2mk+2$. In comparison, our constructions for fault-tolerant base-2 de Bruijn graphs have $N+k$ nodes and degree $4k+4$ and our constructions for base-m de Bruijn graphs have $N+k$ nodes and degree $4(m-1)k+2m$. Thus, our constructions use far fewer nodes and yet have only slightly larger degrees.

The problem of designing fault-tolerant networks for the point-to-point shuffle-exchange networks was addressed by Kuo and Fuchs [10]. However, their approach adds switches between the processors, so their results cannot be directly compared with ours. For the fault-tolerant shuffle-exchange network, we simply use the recent result that a shuffle-exchange network is a subgraph of a base-2 de Bruijn graph of the same size [7]. Thus, the fault-tolerant graph for a shuffle-exchange network, which tolerates up to k node faults, also has a degree $4k+4$. It should be noted that applying the technique of the fault-tolerant de Bruijn graph to the shuffle-exchange network with a natural labeling will yield a graph of degree $6k+4$.

Although in all cases, our constructions have smaller degrees than all previously known constructions with the same properties, the degrees of our constructions are quite large, except for very small values of k. However, it is our hope that the techniques developed here will eventually lead to the development of practical fault-tolerant architectures. In particular, we show how buses can be used to reduce the degrees of the fault-tolerant constructions by almost a factor of 2. It is possible that other techniques, such as adding more than k spare nodes, could be used to reduce the degrees still further.

The remainder of this paper is organized as follows. Notation and definitions are presented in Section II. Fault-tolerant base-2 and base-m de Bruijn networks are given in Sections III and IV, respectively. Section V discusses the use of buses to create fault-tolerant architectures with smaller degrees. Some conclusions and directions for future research are given in Section VI.

II. Notation and Definitions

Given a non-negative integer x, the h-digit base-m representation of x will be written as $[x_{h-1}, x_{h-2}, \cdots, x_0]_m$. Given a set of integers S and an element $x \in S$, the *rank of* x *in* S, denoted *Rank*(x, S), is the number of elements in S that are smaller than x. For example, if S is finite *Rank*$(\min(S), S) = 0$ and *Rank*$(\max(S), S) = |S|-1$. The function $\lambda(x, m, r, s) \stackrel{\text{def}}{=} (xm + r) \bmod s$ will be used throughout this paper.

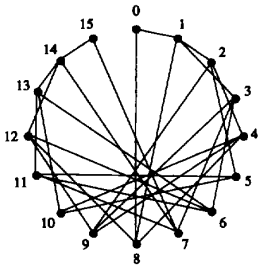

Fig. 1. An example of the base-2 four-digit de Bruijn graph $\mathcal{B}_{2,4}$.

Given a graph G, the set of nodes (vertices) of G will be denoted $V(G)$ and the set of edges of G will be denoted $E(G)$. Self-loops (edges that connect a node to itself) will not be allowed in graphs. (For the sake of simplicity, some graphs will be defined to have connections from nodes to themselves; these self-loops should be ignored.) The *degree of a node* is the number of edges incident with the node and the *degree of a graph* is the maximum of the degrees of its nodes. A graph H is a *subgraph* of a graph G if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. Given a graph G and a set of nodes $W \subseteq V(G)$, the *subgraph of* G *induced by* W is the graph H, where $V(H) = W$ and $E(H) = \{(x, y) \in E(G) \mid x \in W \text{ and } y \in W\}$.

Given graphs G and G', an *embedding of* G *into* G' is a 1-to-1 function $\phi : V(G) \to V(G')$ such that for each edge $(x, y) \in E(G)$, there exists an edge $(\phi(x), \phi(y)) \in E(G')$.

Let G and G' be graphs, and let k be a non-negative integer. We will say that G' is (k, G)-*tolerant* if G' has the following property:

- Let $W \subseteq V(G')$ be an arbitrary set of $|V(G')| - k$ nodes in G', and let H be the subgraph of G' induced by W. Then there exists an embedding of G into H.

III. Base-2 de Bruijn Graphs

In this section, we consider the creation of graphs that can sustain node faults and still be guaranteed to contain a nonfaulty base-2 de Bruijn graph. The base-2 h-digit de Bruijn graph, denoted $\mathcal{B}_{2,h}$, contains 2^h nodes, each of which is labeled with a unique h-bit binary number. It is a degree–4 graph in which there is a link between nodes x and y iff either the last $h-1$ bits of x equal the first $h-1$ bits of y or the first $h-1$ bits of x equal the last $h-1$ bits of y. More formally, given an integer $h \geq 3$, $\mathcal{B}_{2,h}$ has 2^h nodes labeled $0, 1, \cdots, 2^h - 1$. Each node $x = [x_{h-1}, x_{h-2}, \cdots, x_0]_2$ is connected to nodes

$$[x_{h-2}, x_{h-3}, \cdots, x_0, 0]_2,$$
$$[x_{h-2}, x_{h-3}, \cdots, x_0, 1]_2,$$
$$[0, x_{h-1}, x_{h-2}, \cdots, x_1]_2,$$

and

$$[1, x_{h-1}, x_{h-2}, \cdots, x_1]_2.$$

Although it is very natural to define the edges in $\mathcal{B}_{2,h}$ in terms of the binary representations of the nodes, it turns out that a different definition will be more useful for obtaining a fault-tolerant de Bruijn graph. Specifically, given any two distinct nodes x and y in $V(\mathcal{B}_{2,h})$, the pair (x, y) is an edge in $E(\mathcal{B}_{2,h})$ iff there exists an $r \in \{0, 1\}$ such that either $y = \lambda(x, 2, r, 2^h)$ or $x = \lambda(y, 2, r, 2^h)$. It is easily verified that this definition of $\mathcal{B}_{2,h}$ is equivalent to the previous definition. Fig. 1 shows an example of the graph $\mathcal{B}_{2,4}$.

Given this definition of $\mathcal{B}_{2,h}$, we will create a $(k, \mathcal{B}_{2,h})$-tolerant graph, which we denote $\mathcal{B}^k_{2,h}$. Before presenting this fault-tolerant graph, we first describe the reconfiguration algorithm that will be

used to locate the healthy target graph in the fault-tolerant graph with k node faults. We then study which edges have to be present in the fault-tolerant graph, given that this reconfiguration algorithm will be used. Finally, we give a formal definition of the fault-tolerant graph and we prove that it is in fact able to tolerate any k node faults.

A. The Reconfiguration Algorithm

The graph $\mathcal{B}^k_{2,h}$ has $2^h + k$ nodes, numbered $0, 1, \cdots, 2^h + k - 1$. Given any set of k faulty nodes in $\mathcal{B}^k_{2,h}$, the reconfiguration algorithm maps the 2^h nodes in $\mathcal{B}_{2,h}$ to the 2^h nonfaulty nodes in $\mathcal{B}^k_{2,h}$ in a monotonic manner, with each node x of $\mathcal{B}_{2,h}$ being mapped to the $(x+1)$st nonfaulty node in $\mathcal{B}^k_{2,h}$. For example, node 0 is mapped to the first nonfaulty node, and node $2^h - 1$ is mapped to the last nonfaulty node. We use ϕ to denote this mapping from the nodes in $\mathcal{B}_{2,h}$ to the nonfaulty nodes in $\mathcal{B}^k_{2,h}$.

A careful analysis of the structure of $\mathcal{B}_{2,h}$ reveals that $\mathcal{B}^k_{2,h}$ has a degree that is $\Theta(k)$. To see this, consider an arbitrary nonfaulty node a in $\mathcal{B}^k_{2,h}$. The embedding function ϕ will map some node x in $\mathcal{B}_{2,h}$ to a, where $x \in \{a-k, a-k+1, \cdots, a\}$. Now consider the edge (x, y) in $\mathcal{B}_{2,h}$, where $y = \lambda(x, 2, 0, 2^h)$. Given this value of y, there are $k+1$ possible values of $\phi(y)$, namely $\{y, y+1, \cdots, y+k\}$. Thus, in order to guarantee that $(\phi(x), \phi(y)) = (a, \phi(y))$ appears in $\mathcal{B}^k_{2,h}$, we can create edges of the form (a, b) for all nodes b where $b \in \{y, y+1, \cdots, y+k\}$ for some possible value of y. But note that $x \in \{a-k, a-k+1, \cdots, a\}$, so $2x \in \{2a-2k, 2a-2k+1, \cdots, 2a\}$, $y \in \{(2a-2k) \bmod 2^h, (2a-2k+1) \bmod 2^h, \cdots, 2a \bmod 2^h\}$, and $b \in \{(2a-2k) \bmod 2^h, (2a-2k+1) \bmod 2^h, \cdots, (2a+k) \bmod 2^h\}$. Thus, there are only $O(k)$ possible value of b, and we must create only $O(k)$ edges incident to node a. Of course, we have considered only one possibility, namely, an edge (x, y) in $\mathcal{B}_{2,h}$, where $y = \lambda(x, 2, 0, 2^h)$ and $\phi(x) = a$. However, an analogous argument can be used for the other possibilities to show that $\mathcal{B}^k_{2,h}$ need have only a degree of $O(k)$. In fact, a more detailed analysis will be used to improve the constants in the degree of $\mathcal{B}^k_{2,h}$. A formal definition of the graph $\mathcal{B}^k_{2,h}$ and a proof that it is $(k, \mathcal{B}_{2,h})$-tolerant is given next.

B. Fault-Tolerant Graph Definition

For any integer $h \geq 3$ and any integer $k \geq 0$, let $\mathcal{B}^k_{2,h}$ be the graph where the nodes $V(\mathcal{B}^k_{2,h})$ are $\{0, 1, \cdots, 2^h+k-1\}$ and where (x, y) is an edge iff there exists an $r \in \{-k, -k+1, \cdots, k+1\}$ such that either $y = \lambda(x, 2, r, 2^h + k)$ or $x = \lambda(y, 2, r, 2^h + k)$.

The fault-tolerant graph $\mathcal{B}^k_{2,h}$ has a structure that is very similar to that of the target graph $\mathcal{B}_{2,h}$. The only differences are that the fault-tolerant graph has $N + k$ nodes (so all calculations are performed modulo $N + k$) and each node is connected to a block of $2k + 2$ consecutive nodes rather than to a block of 2 consecutive nodes. In particular, note that $\mathcal{B}^0_{2,h} \equiv \mathcal{B}_{2,h}$. Also, note that $\mathcal{B}^k_{2,h}$ contains $2^h + k$ nodes, and that it has degree at most $4k + 4$. Fig. 2 shows an example of the graph $\mathcal{B}^1_{2,4}$.

We now prove that the graph $\mathcal{B}^k_{2,h}$ is in fact $(k, \mathcal{B}_{2,h})$-tolerant. The proof relies on the following technical lemmas.

Lemma 1: Let T be a finite set of integers and let a and b be members of T where $a < b$. If $\delta_a \stackrel{\text{def}}{=} a - \mathit{Rank}(a, T)$ and $\delta_b \stackrel{\text{def}}{=} b - \mathit{Rank}(b, T)$, then $\delta_a \leq \delta_b$.

Proof: Let $U = \{x \in T \mid a < x \leq b\}$. Note that $|U| \leq b - a$ and $\mathit{Rank}(b, T) - \mathit{Rank}(a, T) = |U|$. Therefore, $\delta_b - \delta_a = (b-a) - (\mathit{Rank}(b, T) - \mathit{Rank}(a, T)) \geq |U| - |U| = 0$, so $\delta_a \leq \delta_b$. □

The next technical lemma shows that each edge in $\mathcal{B}_{2,h}$ "wraps around" at most once.

Lemma 2: Let (x, y) be an arbitrary edge in the graph $\mathcal{B}_{2,h}$ and assume, without loss of generality, that there exists an integer $r \in \{0, 1\}$ such that $y = \lambda(x, 2, r, 2^h)$. Then, either $x < y$ and $y = 2x + r$, or $x > y$ and $y = 2x + r - 2^h$.

Proof: Clearly, $x \neq y$, because (x, y) is an edge in a graph. First, consider the case where $x < y$, and assume, for the sake of contradiction, that $y \neq 2x + r$. Then, $y = (2x + r) \bmod 2^h$ and $2x + r \geq 0$, so $2x + r \geq y + 2^h > x + 2^h$, which implies that $x + r > 2^h$. But $x \leq 2^h - 1$ and $r \leq 1$, so $x + r \leq 2^h$, which is a contradiction.

Now consider the case where $x > y$, and assume for the sake of contradiction that $y \neq 2x + r - 2^h$. Because $x \geq 0$ and $r \geq 0$, it follows that $2x + r \geq x > y$. Therefore, $y = (2x + r) \bmod 2^h$, $2x + r > y$, and $2x + r \neq y + 2^h$, which implies that $2x + r \geq y + 2^{h+1} \geq 2^{h+1}$. But $x \leq 2^h - 1$ and $r \leq 1$, so $2x + r \leq 2^{h+1} - 1$, which is a contradiction. □

Theorem 1: For any integer $h \geq 3$ and any integer $k \geq 0$, the graph $\mathcal{B}^k_{2,h}$ is $(k, \mathcal{B}_{2,h})$-tolerant.

Proof: Let $W \subseteq V(\mathcal{B}^k_{2,h})$ be an arbitrary set of 2^h nodes in $\mathcal{B}^k_{2,h}$, and let H be the subgraph of $\mathcal{B}^k_{2,h}$ induced by W. We will show that there exists an embedding of $\mathcal{B}_{2,h}$ into H.

Define the function $\phi : V(\mathcal{B}_{2,h}) \rightarrow V(H)$ such that for each $x \in V(\mathcal{B}_{2,h})$, $\phi(x) \stackrel{\text{def}}{=} z$, where $\mathit{Rank}(z, V(H)) = x$. Note that ϕ is the same monotonically increasing 1-to-1 function that was defined by the reconfiguration algorithm given above. Let (x, y) be an arbitrary edge in $\mathcal{B}_{2,h}$, and assume, without loss of generality, that there exists an integer $r \in \{0, 1\}$ such that $y = \lambda(x, 2, r, 2^h)$. Let $\delta_x \stackrel{\text{def}}{=} \phi(x) - x$ and $\delta_y \stackrel{\text{def}}{=} \phi(y) - y$. Note that $0 \leq \delta_x \leq k$ and $0 \leq \delta_y \leq k$. Let $S \stackrel{\text{def}}{=} \{-k, -k+1, \cdots, k+1\}$. There are two cases:

Case 1: $x < y$. In this case, it follows from Lemma 2 that $y = 2x + r$. Also, $\phi(x) < \phi(y)$, so it follows from Lemma 1 that $\delta_x \leq \delta_y$. Let $s \stackrel{\text{def}}{=} r + \delta_y - 2\delta_x$. Then, $\phi(y) = y + \delta_y = 2x + r + \delta_y = 2(\phi(x) - \delta_x) + r + \delta_y = 2\phi(x) + s$. Note that $0 \leq \phi(y) < 2^h + k$, so $\phi(y) = \lambda(\phi(x), 2, s, 2^h + k)$. But $-k \leq \delta_y - 2\delta_x \leq k$, so $s \in S$ and $(\phi(x), \phi(y))$ is an edge in H. As a result, ϕ is an embedding of $\mathcal{B}_{2,h}$ into H in this case.

Case 2: $x > y$. In this case, it follows from Lemma 2 that $y = 2x + r - 2^h$. Also, $\phi(x) > \phi(y)$, so it follows from Lemma 1 that $\delta_x \geq \delta_y$. Let $s \stackrel{\text{def}}{=} r + \delta_y - 2\delta_x + k$. Then, $\phi(y) = y + \delta_y = 2x + r - 2^h + \delta_y = 2(\phi(x) - \delta_x) + r + \delta_y + k - (2^h + k) = 2\phi(x) + s - (2^h + k)$. Note that $0 \leq \phi(y) < 2^h + k$, so $\phi(y) = \lambda(\phi(x), 2, s, 2^h + k)$. But $-k \leq \delta_y - 2\delta_x + k \leq k$, so $s \in S$ and $(\phi(x), \phi(y))$ is an edge in H. As a result, ϕ is an embedding of $\mathcal{B}_{2,h}$ into H in this case. □

Corollary 1: For any integer $h \geq 3$ and any integer $k \geq 0$, the graph $\mathcal{B}^k_{2,h}$ is $(k, \mathcal{B}_{2,h})$-tolerant and has $2^h + k$ nodes and degree at most $4k + 4$.

Corollary 2: For any integer $h \geq 3$, the graph $\mathcal{B}^1_{2,h}$ is $(1, \mathcal{B}_{2,h})$-tolerant and has $2^h + 1$ nodes and degree at most 8.

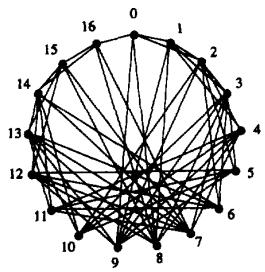

Fig. 2. An example of the graph $\mathcal{B}^1_{2,4}$.

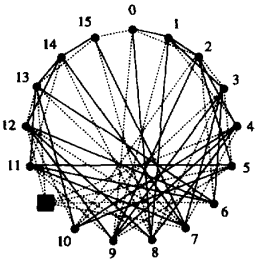

Fig. 3. An example of the new labels of $\mathcal{B}^1_{2,4}$ after one fault.

Fig. 3 shows an example of the new labels of the graph $\mathcal{B}^1_{2,4}$ after one fault. The solid lines in the figure denote the edges that are used after reconfiguration.

IV. Base-m de Bruijn Graphs

The base-2 de Bruijn graph studied in the previous section can be easily generalized to other bases. The base-m h-digit de Bruijn graph, denoted $\mathcal{B}_{m,h}$, contains m^h nodes, each of which is labeled with a unique h-digit base-m number. There is a link between nodes x and y iff either the last $h-1$ digits of x equal the first $h-1$ digits of y or the first $h-1$ digits of x equal the last $h-1$ digits of y. More formally, given integers $m \geq 2$ and $h \geq 3$, $\mathcal{B}_{m,h}$ has m^h nodes labeled $0, 1, \cdots, m^h - 1$. Each node $x = [x_{h-1}, x_{h-2}, \cdots, x_0]_m$ is connected to all nodes of the following form:

$$[x_{h-2}, x_{h-3}, \cdots, x_0, r]_m,$$

and

$$[r, x_{h-1}, x_{h-2}, \cdots, x_1]_m,$$

where $r \in \{0, 1, \cdots, m-1\}$.

As was the case with the base-2 de Bruijn graph, it turns out that a different definition of the base-m de Bruijn graph will be more useful than the above definition in obtaining a fault-tolerant graph. Specifically, the pair (x, y) is an edge in $\mathcal{B}_{m,h}$ iff there exists an $r \in \{0, 1, \cdots, m-1\}$ such that either $y = \lambda(x, m, r, m^h)$ or $x = \lambda(y, m, r, m^h)$. It is easily verified that this definition of $\mathcal{B}_{m,h}$ is equivalent to the previous definition.

The creation of a fault-tolerant base-m de Bruijn graph is a natural extension of the fault-tolerant base-2 de Bruijn graph defined in the previous section. In particular, the reconfiguration algorithm for the base-m de Bruijn graph is identical to the one used for the base-2 de Bruijn graph and will not be repeated. A formal definition of the fault-tolerant base-m de Bruijn graph and a proof of its properties is given next. The proof relies on a lemma, which is proven first.

A. Fault-Tolerant Graph Definition

For any integers $m \geq 2$, $h \geq 3$ and $k \geq 0$, let $\mathcal{B}^k_{m,h}$ be the graph where the nodes $V(\mathcal{B}^k_{m,h})$ are $\{0, 1, \cdots, m^h + k - 1\}$ and where (x, y) is an edge iff there exists an $r \in \{(m-1)(-k), (m-1)(-k)+1, \cdots, (m-1)(k+1)\}$ such that either $y = \lambda(x, m, r, m^h + k)$ or $x = \lambda(y, m, r, m^h + k)$.

Note that $\mathcal{B}^0_{m,h} \equiv \mathcal{B}_{m,h}$. Also, note that $\mathcal{B}^k_{m,h}$ contains $m^h + k$ nodes and it has degree at most $(m-1)(4k) + 2m$.

Lemma 3: Let (x, y) be an arbitrary edge in the graph $\mathcal{B}_{m,h}$ and assume, without loss of generality, that there exists an integer $r \in \{0, 1, \cdots, m-1\}$ such that $y = \lambda(x, m, r, m^h)$, and let t be an integer such that $y = mx + r - tm^h$. Then, either $x < y$ and $t \in \{0, 1, \cdots, m-2\}$ or $x > y$ and $t \in \{1, 2, \cdots, m-1\}$.

Proof: Clearly, $x \neq y$, because (x, y) is an edge in a graph. Note that $mx + r \geq 0$ and $y < m^h$, so $t > -1$. Also, note that $y \geq 0$, $x \leq m^h - 1$ and $r < m$, so $mx + r < m^{h+1}$ and $t < m$. Thus, $t \in \{0, 1, \cdots, m-1\}$.

Now consider the case where $x < y$, and assume, for the sake of contradiction, that $t = m-1$. Then, $x \leq m^h - 2$ and $x < y = mx + r - (m-1)m^h$, so $0 < (m-1)x + r - (m-1)m^h \leq (m-1)(-2) + r$ and $r > 2m - 2 \geq m$, which is a contradiction.

Now consider the case where $x > y$, and assume, for the sake of contradiction, that $t = 0$. Because $m \geq 2$, $x \geq 0$, and $r \geq 0$, it follows that $mx + r \geq x > y$. But $y = mx + r$, which is a contradiction. □

Theorem 2: For any integers $m \geq 2$, $h \geq 3$, and $k \geq 0$, the graph $\mathcal{B}^k_{m,h}$ is $(k, \mathcal{B}_{m,h})$-tolerant.

Proof: Let $W \subseteq V(\mathcal{B}^k_{m,h})$ be an arbitrary set of m^h nodes in $\mathcal{B}^k_{m,h}$, and let H be the subgraph of $\mathcal{B}^k_{m,h}$ induced by W. We will show that there exists an embedding of $\mathcal{B}_{m,h}$ into H.

Let (x, y) be an arbitrary edge in $\mathcal{B}_{m,h}$, and assume, without loss of generality, that there exists an integer $r \in \{0, 1, \cdots, m-1\}$ such that $y = \lambda(x, m, r, m^h)$, and let t be an integer such that $y = mx + r - tm^h$. Define the function $\phi : V(\mathcal{B}_{m,h}) \rightarrow V(H)$ such that for each $x \in V(\mathcal{B}_{m,h})$, $\phi(x) \stackrel{\text{def}}{=} z$, where $Rank(z, V(H)) = x$. Note that ϕ is a monotonically increasing 1-to-1 function. Let $\delta_x \stackrel{\text{def}}{=} \phi(x) - x$ and $\delta_y \stackrel{\text{def}}{=} \phi(y) - y$. Note that $0 \leq \delta_x \leq k$ and $0 \leq \delta_y \leq k$. Let $S \stackrel{\text{def}}{=} \{(m-1)(-k), (m-1)(-k)+1, \cdots, (m-1)(k+1)\}$, and let $s \stackrel{\text{def}}{=} kt + r + \delta_y - m\delta_x$. Then,

$$\begin{aligned}
\phi(y) &= y + \delta_y \\
&= mx + r - tm^h + \delta_y \\
&= m(\phi(x) - \delta_x) + r - t(m^h + k - k) + \delta_y \\
&= m\phi(x) + kt + r + \delta_y - m\delta_x - t(m^h + k) \\
&= m\phi(x) + s - t(m^h + k).
\end{aligned}$$

Note that $0 \leq \phi(y) < m^h + k$, so $\phi(y) = \lambda(\phi(x), m, s, m^h + k)$. There are two cases.

- *Case 1:* $x < y$. In this case, it follows from Lemma 3 that $t \in \{0, 1, \cdots, m-2\}$. Also, $\phi(x) < \phi(y)$, so it follows from Lemma 1 that $\delta_x \leq \delta_y$. But $s \geq \delta_y - m\delta_x \geq (1-m)\delta_x \geq (m-1)(-k)$ and $s \leq k(m-2) + (m-1) + k = (m-1)(k+1)$, so $s \in S$ and $(\phi(x), \phi(y))$ is an edge in H. As a result, ϕ is an embedding of $\mathcal{B}_{m,h}$ into H in this case.
- *Case 2:* $x > y$. In this case, it follows from Lemma 3 that $t \in \{1, 2, \cdots, m-1\}$. Also, $\phi(x) > \phi(y)$, so it follows from Lemma 1 that $\delta_x \geq \delta_y$. But $s \geq k - mk = (m-1)(-k)$ and $s \leq k(m-1) + (m-1) + \delta_x - m\delta_x \leq k(m-1) + (m-1) = (m-1)(k+1)$, so $s \in S$ and $(\phi(x), \phi(y))$ is an edge in H. As a result, ϕ is an embedding of $\mathcal{B}_{m,h}$ into H in this case.

□

Corollary 3: For any integers $m \geq 2$, $h \geq 3$, and $k \geq 0$, the graph $\mathcal{B}^k_{m,h}$ is $(k, \mathcal{B}_{m,h})$-tolerant and has $m^h + k$ nodes and degree at most $(m-1)(4k) + 2m$.

Corollary 4: For any integers $m \geq 2$ and $h \geq 3$, the graph $\mathcal{B}^1_{m,h}$ is $(1, \mathcal{B}_{m,h})$-tolerant and has $m^h + 1$ nodes and degree at most $6m - 4$.

V. Implementations with Buses

Up to this point, we have considered only fault-tolerant architectures that use point-to-point connections between pairs of processors (and thus can be viewed as graphs). In this section, we show how fault-tolerant architectures with smaller degrees can be obtained by using buses. Buses can be used to reduce the degrees of all of the constructions that have been presented. In order to simplify the presentation, however, we consider only the case of base-2 de Bruijn graphs here.

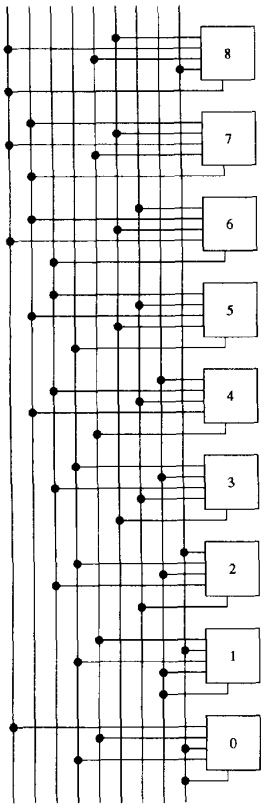

Fig. 4. An example of the graph $\mathcal{B}^1_{2,3}$ using bus implementation.

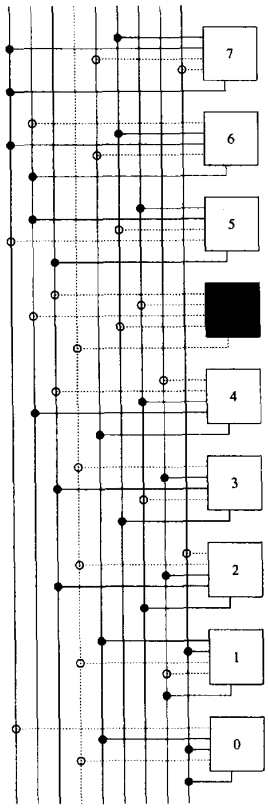

Fig. 5. An example of the reconfiguration after one fault in thegraph $\mathcal{B}^1_{2,3}$ using bus implementation.

Recall that in the base-2 de Bruijn graph $\mathcal{B}_{2,h}$, each node i is connected to the pair of nodes $2i \bmod 2^h$ and $(2i+1) \bmod 2^h$. Therefore, all of the connectivity of the graph $\mathcal{B}_{2,h}$ will be maintained if each such pair of edges is replaced with a single bus that connects node i to both node $2i \bmod 2^h$ and $(2i+1) \bmod 2^h$. Because only a single value can be transmitted over the bus in unit time, the implementation using buses will be approximately a factor of 2 slower than the implementation with direct connections, assuming that the latter implementation allows two different values to be sent from a single processor in unit time. If only one value can be sent from a processor in unit time, however, then little or no slowdown is incurred by using buses. The reason that a slight slowdown may be incurred, even when each processor can send only one value at a time is that the bus may have a larger capacitance than either of the direct connections that it replaces. However, the exact relationship between the capacitance of the bus and the capacitances of the direct connections depends on the geometry of the layout and is beyond the scope of this paper.

The same technique can be used to reduce the degrees of the fault-tolerant graphs. Recall that in the fault-tolerant graph $\mathcal{B}^k_{2,h}$, each node i is connected to a block of $2k+2$ consecutive nodes beginning with node $(2i-k) \bmod (2^h+k)$. Therefore, a single bus could be used to connect each node i to all $2k+2$ of these nodes. Once again, this construction will incur a slowdown by approximately a factor of 2 if each processor can send two different values in unit time, and little or no slowdown if each processor can send only one value in unit time. This use of buses results in a fault-tolerant architecture with degree $2k+3$. Fig. 4 shows an example of the implementation of the fault-tolerant graph $\mathcal{B}^1_{2,3}$ using buses. Fig. 5 shows an example of the reconfiguration after one fault in the graph $\mathcal{B}^1_{2,3}$ using bus implementation.

Finally, note that if the bus connecting node i to the block of $2k+2$ consecutive nodes beginning with node $(2i-k) \bmod (2^h+k)$ is faulty, we can avoid using the bus by treating node i as being faulty. Thus, even bus faults can be tolerated efficiently with this architecture. Note that this is possible only because we use the bus in a restrictive way (in which node i is always being connected to another node on the bus), so treating node i as being faulty will prevent the bus from being used. In contrast, if a bus that connects p nodes could be used to implement a connection between any pair of the p nodes, one would have to treat $p-1$ of the nodes as being faulty in order to guarantee that the bus will not be used.

VI. Conclusion

This paper has shown constructions for adding fault tolerance to de Bruijn, which easily implies a construction for shuffle-exchange network [7]. All of these constructions have the minimum number of nodes required for the given level of fault tolerance. Also, all of the constructions have a smaller degree than any previously known constructions with the same fault-tolerance properties. In particular, all of the constructions have a degree that is linear in the number of faults tolerated.

Although the given constructions have smaller degrees than any previously known constructions with the same properties, they are practical only for very small values of k. One technique for reducing the degrees of the fault-tolerant graphs is to use buses in the manner

that was described in Section V. It is possible that other techniques, such as creating fault-tolerant graphs with more than $N+k$ nodes, can be used to reduce the degrees still further. Also, it has not been proven that the given constructions have the smallest possible degrees. As a result, it would be interesting to prove lower bounds on the degrees of graphs with the given fault-tolerance properties.

REFERENCES

[1] J.-C. Bermond and C. Peyrat, "de Bruijn and Kautz networks: A competitor for the hypercube?" in F. André and J. P. Verjus, Eds., *Hypercube and Distributed Computers*. Amsterdam: Elsevier B.V. (North-Holland), 1989, pp. 279–293.

[2] J. Bruck, R. Cypher, and C.-T. Ho, "On the construction of fault-tolerant cube-connected cycles networks," *Proc. 1991 Int. Conf. Parallel Processing*, vol. I, pp. 692–693.

[3] ——, "Fault-tolerant meshes with minimal numbers of spares," *Proc. 1991 Symp. Parallel Distrib. Processing*, pp. 288–295.

[4] S. Dutt and J. P. Hayes, "On designing and reconfiguring k-fault-tolerant tree architectures," *IEEE Trans. Comput.*, vol. C-39, pp. 490–503, Apr. 1990.

[5] ——, "Designing fault-tolerant systems using automorphisms," *J. Parallel Distrib. Computing*, vol. 12, pp. 249–268, 1991.

[6] ——, "Some practical issues in the design of fault-tolerant multiprocessors," *Proc. 21st Int. Symp. Fault-Tolerant Computing*, 1991, pp. 292–299.

[7] R. Feldmann and W. Unger, "The cube-connected cycles network is a subgraph of the butterfly network," *Parallel Processing Lett.*, vol. 2, pp. 13–19, Mar. 1992.

[8] A.-H. Esfahanian and S. L. Hakimi, "Fault-tolerant routing in de Bruijn communication networks," *IEEE Trans. Comput.*, vol. C-34, no. 9, pp. 777–788, Sept. 1985.

[9] J. P. Hayes, "A graph model for fault-tolerant computing systems," *IEEE Trans. Comput.*, vol. C-25, no. 9, pp. 875–884, Sept. 1976.

[10] S.-Y. Kuo and W. K. Fuchs, "Reconfigurable cube-connected cycles architectures." *J. Parallel Distrib. Computing*, vol. 9, pp. 1–10, 1990.

[11] F. P. Preparata and J. Vuillemin, "The cube-connected cycles: A versatile network for parallel computation," *Commun. ACM*, vol. 24, no. 5, pp. 300–309, May 1981.

[12] M. R. Samatham and D. K. Pradhan, "The de Bruijn multiprocessor network: A versatile parallel processing and sorting network for VLSI," *IEEE Trans. Comput.*, vol. 38, pp. 567–581, Apr. 1989.

[13] H. S. Stone, "Parallel processing with the perfect shuffle," *IEEE Trans. Comput.*, vol. C-20, pp. 153–161, Feb. 1971.

Memory Bandwidth Analysis of Hierarchical Multiprocessors Using Model Decomposition and Steady-State Flow Analysis

Syed Masud Mahmud and L. Tissa Samaratunga

***Abstract*— For memory bandwidth analysis, most of the time the researchers [1]–[8] discard the requests that are not accepted during a memory cycle. Such an assumption simplifies the analysis and produces negligible discrepancies from the actual results [3] for a system with a non-hierarchical interconnection network. However, the assumption, "*the requests that are not accepted during a memory cycle are discarded*," cannot be used for a multiprocessor system with a Hierarchical Interconnection Network (HIN), because the error introduced due to such an assumption can be several orders of magnitude higher than the actual bandwidth! An improved analytical model to determine the bandwidth of a HIN-based system is presented in this short note.**

***Index Terms*—Bandwidth, crossbar, probabilistic model, cluster-based system, hierarchical multiprocessor, hierarchical requesting model, uniform reference, steady-state flow approach, performance analysis**

I. INTRODUCTION

In order to develop a bandwidth analysis model, most of the time the researchers [1]–[8] use the assumption, "*the requests that are not accepted during a memory cycle are discarded*". Such an assumption simplifies the analysis and produces negligible discrepancies from the actual results [3] for a system with a non-hierarchical interconnection network, but it can introduce misleading conclusion about the performance of a system with a hierarchical interconnection network (HIN). Recently such an analysis for a system with a HIN appeared in the literature [10], and the results presented in that paper are way off than they are supposed to be.

This short note shows a more accurate analytical model for determining memory bandwidth for shared memory multiprocessors, where the processors and memory modules are connected by a HIN. The *steady-state flow approach* [9] can be used to develop a more realistic model for bandwidth computation. However, such a technique cannot be applied directly to the model shown in [10]. In our modeling technique we decomposed the memory references of a hierarchical system into as many groups as the number of hierarchy levels in the system. A model is developed for each group of memory references to determine its contribution to the bandwidth. Thus, we have a number of decomposed models rather than a single model for the entire system. The steady-state flow approach technique was applied to every decomposed model to take care of the fact that the requests which are not accepted in a cycle, are resubmitted during the next cycle. The total memory bandwidth is determined by adding the contributions from all the decomposed models. We analyzed the hierarchical multiprocessor system presented in [10] using our modeling technique. A simulation model is also developed to determine the accuracy of our analytical model. The numerical results obtained from our analytical model match closely to those obtained from the simulation model. But the numerical results obtained from the model shown in [10] are several orders of magnitude higher than the simulation results, especially

Manuscript received October 14, 1992; revised May 3, 1993.

The authors are with the Department of Electrical and Computer Engineering, Wayne State University, Detroit, MI 48202. e-mail: smahmud@ece.eng.wayne.edu.

IEEE Log Number 9216785.