Regular Paper

A Method of Fault-tolerant All-to-All Personalized Communication in Banyan Networks

Masashi Yaku† and Hiroshi Masuyama††

In this paper, all-to-all personalized communication for multistage interconnection networks, in particular for banyan networks, is discussed. All-to-all personalized communication is one of the most dense collective communication patterns and occurs in many important applications for parallel computing. Since the communication time required for an all-to-all personalized communication is quite costly, efficient communication schemes are important in order to achieve a high performance. We developed a new tolerable scheme for a single non-critical fault, then presented the upper bounds of required communication time due to the stages with the faulty element.

1. Introduction

All-to-all personalized communication is one of the most dense collective communication patterns. Many schemes have been developed for several parallel computing networks, such as hypercube, mesh, and multistage interconnection networks [1)~10)].

Johnsson and Ho [1)] proposed optimal all-to-all personalized communication algorithms on an n-node hypercube with $O(n \log n)$ and $O(n)$ time complexity for a one-port model and a $\log n$-port model, respectively. Typical all-to-all personalized communication algorithms on a two-dimensional mesh and torus have time complexity $O(n^{3/2})$, where n is the total number of nodes. Yang and Wang [11)] have developed an all-to-all personalized communication optimum algorithm for banyan networks, given the time complexity $O(n)$. These schemes have aimed mainly at nonfaulty networks.

On the other hand, Park and Bose [12)] proposed a fault-tolerant all-to-all broadcasting algorithm in a $\log n$-dimensional hypercube with up to $\lfloor \log n/2 \rfloor$ faulty links (or faulty nodes), however, it was not a personalized communication algorithm. There has not yet been any developed fault-tolerant all-to-all personalized communication algorithms for any faulty network.

Multistage interconnection networks are a vital component of parallel computing systems, and enable the computing elements to communicate among themselves. The performance of the system depends a great deal on the extent of the interprocessor communication. The failure of a component can bring down the system performance, unless sufficient fault tolerant schemes are provided. Any of the multistage interconnection networks is referred to as a banyan network. In this paper, we will focus on a method of all-to-all personalized communication tolerable for faulty banyan networks, and estimate the upper bound of required communication time.

The rest of the paper is organized as follows. Section 2 summarizes an all-to-all personalized communication scheme in a fault-free case based on Latin square and permutations. Section 3 presents an all-to-all personalized communication scheme in faulty cases introduced from the above scheme. The conclusion follows in Section 4.

2. All-to-All Personalized Communication in Fault-free Cases

In distributed and also shared memory systems, communication among the processors is performed mainly via message passing. Since the communication time may be quite expensive compared to the computation time, efficient communication schemes are extremely important to achieve a high performance in the system. Johnsson and Ho [1)] introduced four different communication primitives:

(1) *one-to-all broadcasting (or single node broadcasting)* in which a single node distributes common data to all other nodes,
(2) *one-to-all personalized communication (or scattering)* in which a single node sends unique data to all other nodes,
(3) *all-to-all broadcasting (or multimode*

† Graduate School, Tottori University
†† Information and Knowledge Engineering, Tottori University

broadcasting) in which all nodes broadcast concurrently to all other nodes, and

(4) *all-to-all personalized communication (or total exchange)* where each and every node sends unique data to every other node.

The last communication primitive is one of the most dense collective communication patterns and occurs in many important applications for parallel computing. The remaining three communication primitives can be viewed as a special case of all-to-all personalized communication. So, Y. Yang and J. Wang developed the last communication primitive algorithm for a $\log n$ stage banyan network presented in Ref. 11). They also presented the number of required cycles to be n, which is optimum. We will develop a method for faulty networks and present the upper bound of the number of required cycles.

A. Latin Square and Permutations

A Latin square is defined as an $n \times n$ Matrix L

$$L = \begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,n-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n-1,0} & a_{n-1,1} & \cdots & a_{n-1,n-1} \end{bmatrix}$$

in which the entries $a_{i,j}$'s are numbers in $\{0, 1, 2, \cdots, n-1\}$ and no two entries in a row (or a column) have the same value.

Let $P = (p_0, p_1, p_2, \cdots, p_{n-1})$ be an ordered sequence whose elements are elements in the original ordered sequence $T = (0, 1, 2, \cdots, n-1)$. A permutation is a conversion from T to P. In other words, a permutation is a bijection (one to one mapping) from $S = \{0, 1, 2, \cdots, n-1\}$ to S. Permutation ρ which maps i to a_i (that is, $\rho(i) = a_i$) is represented by

$$\rho = \begin{pmatrix} 0 & 1 & 2 & \cdots & n-1 \\ a_0 & a_1 & a_2 & \cdots & a_{n-1} \end{pmatrix}$$

where $a_i \neq a_j$ for $i \neq j$. We refer to permutation $a_i = i$ for all i as an identity permutation I. The reverse permutation of ρ is denoted as ρ^{-1}.

The banyan network considered in this section is an $n \times n$ network composed of $m = \log n$ stages of 2×2 switches as shown as an example of $n = 8$ in **Fig. 1**. E_{12} means the 2-nd switching element on the 1-st stage.

Let $\sigma_i (0 \leq i \leq m-1)$ denote the stage permutation realized by a set of $n/2$ switches on stage i, and $\tau_j (0 \leq j \leq m-2)$ denote the interstage permutation realized by the set of interstage links between stages j and $j+1$. Permutations τ_0 and τ_1 for an 8×8 banyan network are

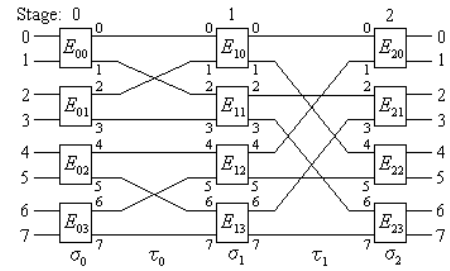


Fig. 1 An 8×8 banyan network composed of 2×2 switches.

$$\tau_0 = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 0 & 2 & 1 & 3 & 4 & 6 & 5 & 7 \end{pmatrix},$$

$$\tau_1 = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 0 & 4 & 2 & 6 & 1 & 5 & 3 & 7 \end{pmatrix}.$$

In general, permutation τ_j can be expressed also by the following 2 bits permute permutation T_j;

$$T_j = \begin{pmatrix} p_{m-1}p_{m-2}\cdots p_{j+2}p_{j+1}p_j \cdots p_1 \ \ p_0 \\ p_{m-1}p_{m-2}\cdots p_{j+2} \ \ p_0 \ \ p_j \cdots p_1 p_{j+1} \end{pmatrix}$$

where $p_{m-1}p_{m-2}\cdots p_1 p_0$ is the binary representation of any element of $\{0, 1, \cdots, n-1\}$.

Clearly, a one-to-one mapping from the network inputs to the outputs is an admissible permutation for the banyan network. An admissible permutation for a banyan network can be expressed by a composition of m stage permutations and $(m-1)$ interstage permutations. The number of all admissible permutations for a banyan network is given as $(2^{n/2})^m = n^{n/2}$ because of a total $2^{n/2}$ possible choices for each nonfixed σ_i. In the next section, we will show that Latin square L can be obtained by taking either of the following 2 permutations I and σ as σ_i for all i (This means only 2^m of $n^{n/2}$ admissible permutations are used for realizing L, and developing an all-to-all personalized communication algorithm is based on a Latin square whereby each row corresponds to an admissible permutation of the banyan networks);

$$I = \begin{pmatrix} 0\,1\,2\,3\cdots & i & i+1\cdots n-2\,n-1 \\ 0\,1\,2\,3\cdots & i & i+1\cdots n-2\,n-1 \end{pmatrix},$$

$$\sigma = \begin{pmatrix} 0\,1\,2\,3\cdots & i & i+1\cdots n-2\,n-1 \\ 1\,0\,3\,2\cdots i+1 & i & \cdots n-1\,n-2 \end{pmatrix},$$

$i =$ any even integer.

Permutation σ can be expressed also by one bit complement permutation Σ;

$$\Sigma = \begin{pmatrix} p_{m-1}p_{m-2}\cdots p_{j+2}p_{j+1}p_j \cdots p_1 p_0 \\ p_{m-1}p_{m-2}\cdots p_{j+2}p_{j+1}p_j \cdots p_1 \bar{p}_0 \end{pmatrix}.$$

B. Communication Algorithm

We will first define a product $\alpha \cdot \beta$ of two permutations α and β, as follows; product $\alpha \cdot \beta$ is a permutation $i \to q_i$ when α and β are permutations $i \to p_i$ and $p_i \to q_i$, respectively. For example,

$$\begin{aligned}\alpha \cdot \beta &= \begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 1 & 3 & 4 & 0 & 2 \end{pmatrix} \cdot \begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 2 & 4 & 0 & 3 & 1 \end{pmatrix} \\ &= \begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 1 & 3 & 4 & 0 & 2 \end{pmatrix} \cdot \begin{pmatrix} 1 & 3 & 4 & 0 & 2 \\ 4 & 3 & 1 & 2 & 0 \end{pmatrix} \\ &= \begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 4 & 3 & 1 & 2 & 0 \end{pmatrix}\end{aligned}$$

Product of permutations is not commutative, that is, $\alpha \cdot \beta \neq \beta \cdot \alpha$ ($\alpha \cdot \beta$ and $\beta \cdot \alpha$ result in different products). A Product of three or more permutations can be obtained inductively from a product of two permutations.

Based on a product of $(2m-1)$ permutations, we can construct Latin squares as described in the following *Algorithm A*.

Algorithm A:

Make 2^m different products of $(2m-1)$ permutations as follows and insert each of them into a different row of $2^m \times 2^m$ matrix L';

$\sigma_0 \cdot \tau_0 \cdot \sigma_1 \cdot \tau_1 \cdot \sigma_2 \cdot \tau_2 \cdots \sigma_{m-2} \cdot \tau_{m-2} \cdot \sigma_{m-1}$

where $\sigma_i (i = 0, 1, 2, \cdots, m-1)$ is σ or I.

Since each row of L' is driven by each different product of $(2m-1)$ permutations, L' doesn't have the same rows. Obviously, no two entries have the same value in a row, and also no two entries have the same value in a column. Then, we are able to obtain the following property.

Property 1 *Matrix L' is a Latin square.*

Example 1 *L' is obtained for an 8×8 matrix as follows:*

$$L' = \begin{array}{c} \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{matrix} \\ \begin{bmatrix} 0 & 2 & 4 & 6 & 1 & 3 & 5 & 7 \\ 1 & 3 & 5 & 7 & 0 & 2 & 4 & 6 \\ 4 & 6 & 0 & 2 & 5 & 7 & 1 & 3 \\ 5 & 7 & 1 & 3 & 4 & 6 & 0 & 2 \\ 2 & 0 & 6 & 4 & 3 & 1 & 7 & 5 \\ 3 & 1 & 7 & 5 & 2 & 0 & 6 & 4 \\ 6 & 4 & 2 & 0 & 7 & 5 & 3 & 1 \\ 7 & 5 & 3 & 1 & 6 & 4 & 2 & 0 \end{bmatrix} \end{array} \begin{array}{l} \leftarrow I \cdot \tau_0 \cdot I \cdot \tau_1 \cdot I \\ \leftarrow I \cdot \tau_0 \cdot I \cdot \tau_1 \cdot \sigma \\ \leftarrow I \cdot \tau_0 \cdot \sigma \cdot \tau_1 \cdot I \\ \leftarrow I \cdot \tau_0 \cdot \sigma \cdot \tau_1 \cdot \sigma \\ \leftarrow \sigma \cdot \tau_0 \cdot I \cdot \tau_1 \cdot I \\ \leftarrow \sigma \cdot \tau_0 \cdot I \cdot \tau_1 \cdot \sigma \\ \leftarrow \sigma \cdot \tau_0 \cdot \sigma \cdot \tau_1 \cdot I \\ \leftarrow \sigma \cdot \tau_0 \cdot \sigma \cdot \tau_1 \cdot \sigma \end{array}$$

Property 2 *All-to-all personalized communication in a $\log n$-stages banyan network can be performed in n cycles.*

Proof: It is obvious from the above discussion.

In the following, we will treat only the matrix L' made in order of permutation products as shown in the above example, as Latin square driven from *Algorithm A*, that is,

$$L' = \begin{bmatrix} the\ 1{-}st\ row \\ the\ 2{-}nd\ row \\ the\ 3{-}rd\ row \\ the\ 4{-}th\ row \\ \vdots \\ the\ (n-1){-}th\ row \\ the\ n{-}th\ row \end{bmatrix} \begin{array}{l} \leftarrow \Gamma_0 \\ \leftarrow \Gamma_1 \\ \leftarrow \Gamma_2 \\ \leftarrow \Gamma_3 \\ \vdots \\ \leftarrow \Gamma_{(n-2)} \\ \leftarrow \Gamma_{(n-1)} \end{array}$$

$$\begin{aligned} \Gamma_0 &= I \cdot \tau_0 \cdot I \cdot \tau_1 \cdot I \cdots \tau_{m-3} \cdot I \cdot \tau_{m-2} \cdot I \\ \Gamma_1 &= I \cdot \tau_0 \cdot I \cdot \tau_1 \cdot I \cdots \tau_{m-3} \cdot I \cdot \tau_{m-2} \cdot \sigma \\ \Gamma_2 &= I \cdot \tau_0 \cdot I \cdot \tau_1 \cdot I \cdots \tau_{m-3} \cdot \sigma \cdot \tau_{m-2} \cdot I \\ \Gamma_3 &= I \cdot \tau_0 \cdot I \cdot \tau_1 \cdot I \cdots \tau_{m-3} \cdot \sigma \cdot \tau_{m-2} \cdot \sigma \\ &\vdots \\ \Gamma_{(n-2)} &= \sigma \cdot \tau_0 \cdot \sigma \cdot \tau_1 \cdot \sigma \cdots \tau_{m-3} \cdot \sigma \cdot \tau_{m-2} \cdot I \\ \Gamma_{(n-1)} &= \sigma \cdot \tau_0 \cdot \sigma \cdot \tau_1 \cdot \sigma \cdots \tau_{m-3} \cdot \sigma \cdot \tau_{m-2} \cdot \sigma \end{aligned}$$

Let L' divide into two $n \times n/2$ matrices L'_0 and L'_1, and further, L'_0 into two $n \times n/4$ matrices L'_{00} and L'_{01}.

$$\begin{aligned} L' &= [L'_0 | L'_1] \\ &= [L'_{00} | L'_{01} | L'_1] \end{aligned}$$

Let *4 6 0 2 5 7 1 3* of the 3-rd row of L' in *Example 1* be called the entry sequence of the 3-rd row, then *1 3 5 7* is the entry sequence of the 2-nd row of L'_0.

Though the following property is obvious by reason of systematic and sequential products of permutation in L', in order to draw reader's attention to the property of L', we will prepare the following property.

Property 3 *Two sets of entry sequences of all L'_0 and L'_1 rows are the same. For the two entry sequences $L'_{00}(j)$ and $L'_{01}(j)$ of the j-th row of L'_{00} and L'_{01}, respectively, the set of elements which compose $L'_{00}(j)$ is one of both subsets of $\{0, 1, \cdots, n/2-1\}$ and $\{n/2, n/2+1, \cdots, n-1\}$, and the set of elements which compose $L'_{01}(j)$ is the other.*

3. All-to-All Personalized Communication in Faulty Cases

Banyan networks possess the property of full access which means that data from any input link can be transferred to any output link in a single pass (we will use "cycle" in this sense) through the network, where a unique path from any input link of the network to any output link is held. However, a problem on the minimization of delivery loss of the information from an input link at the expense of routing

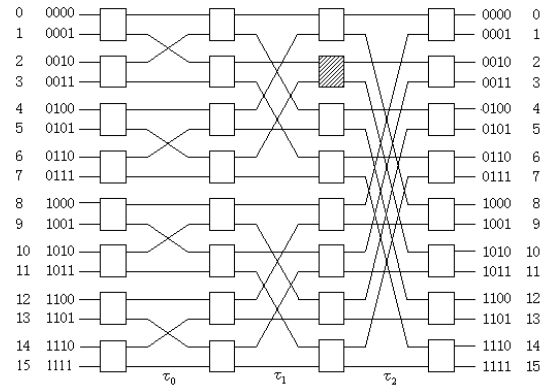
overhead occurs when faults exist. Especially when a banyan network is used for processor-to-processor connection, the effect of the faults on the system can be reduced by allowing multiple passes through the network. The network is said to possess the *dynamic full access* (DFA) capability if every processor in the system can communicate with every other processor in a finite number of cycles through the network, routing the information through intermediate PE's if necessary [13]. Fault-tolerance criterion for banyan networks in this paper is presenting the DFA capability.

The fault model we consider is one in which only the switching elements fail because a faulty link can be accommodated by treating it as a faulty switching element. In addition, we don't consider critical faults by which the DFA capability is destroyed, and in this paper we treat a single fault on inside stages.

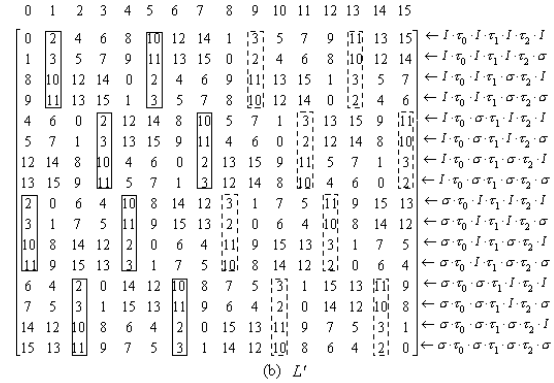
Figure 2 shows a 16×16 banyan network with faulty switching element E_{21} and the matrix L'. In L', for example, 2, 3, 10, and 11 in a square written by solid lines in column 0 means output links which have no path from input link 0 because of faulty switching element E_{21}. All other sequences written by solid lines mean output links which have no path from each input link assigned by the column because of faulty switching element E_{21}, similarly.

The above 4 paths from input link 0 to output links 2, 3, 10,and 11 can be constructed by allowing each two cycles, that is, $0 \to 8 \to 2$ $(0 \to 8, 8 \to 2)$, $0 \to 8 \to 3$, $0 \to 8 \to 10$, and $0 \to 8 \to 11$, respectively. In these 4 routes, the common path $0 \to 8$ means permutation product Γ_2 $(= I \cdot \tau_0 \cdot I \cdot \tau_1 \cdot \sigma \cdot \tau_2 \cdot I)$, and path $8 \to 2$ means Γ_9. The other 4 paths from input link 2 to output links 2, 3, 10, and 11 can also be constructed by allowing each two passes, that is, $2 \to 12 \to 2$ $(2 \to 12, 12 \to 2)$, $2 \to 12 \to 3$, $2 \to 12 \to 10$, and $2 \to 12 \to 11$, respectively. In these 8 routes as we have seen, Γ_2, Γ_8, Γ_9, Γ_{10}, and Γ_{11} are common to pairs of paths $0 \to 8$ and $2 \to 12$, $8 \to 3$ and $12 \to 11$, $8 \to 2$ and $12 \to 10$, $8 \to 11$ and $12 \to 3$, and $8 \to 10$ and $12 \to 2$, respectively. This means the above 8 routes can be constructed by using the following extra 8-permutation-products sequence.

$$\Gamma_2$$
$$\Gamma_8$$
$$\Gamma_2$$
$$\Gamma_9$$



(a) 16×16 banyan network with faulty switching element E_{21}



(b) L'

Fig. 2 (a) A 16×16 banyan network and (b) the matrix L'.

$$\Gamma_2$$
$$\Gamma_{10}$$
$$\Gamma_2$$
$$\Gamma_{11}$$

Observing L' in Fig. 2 (b), we can construct all other 6×4 paths related to input links 1, 3, 4, 5, 6, and 7 can be constructed by allowing routes via two common transitive output terminals 8 and 12. We next consider a case of faulty switching element E_{20}. In this case, fault-tolerant routes can be constructed by allowing routes via another two common transitive output terminals 10 and 14. In the case of faulty switching element E_{2l} $(0 \le l \le n/4 - 1)$, fault-tolerant routes can be constructed by allowing routes via either of the two pairs of common transitive output terminals $(8, 12)$ and $(10, 14)$. On the other hand, in the case of faulty switching element E_{1l}, by observing L', it can be obtained that necessary fault-tolerant routes can be constructed by allowing routes via the same number of common transitive output terminals, that is 2. We can say, by observing L', that the

number of common transitive output terminals discussed above can be taken as always 2 for a faulty switching element on any inside stage.

From the above discussion, in the case of faulty switching element E_{il} $(0 < i \leq \log n - 2, 0 \leq l \leq n/4-1)$, we can perform an all-to-all personalized communication by preparing extra $2n$ cycles. This means total $3n$ cycles are required.

In the above discussion, input links pairs $(0,2),(1,3),(4,6),\cdots$ are the key for the fault-tolerance. We will next consider these input link pairs in a generalized $n \times n$ matrix L'. **Figure 3** shows the matrix L' in the case of faulty switching element E_{il} $(0 < i \leq \log n - 2, 0 \leq l \leq n/4-1)$. In this figure, A and B mean domains where a set of output links which have some disconnected paths from an input link is included and not included, respectively. The key input link pairs can be obtained by the following algorithm.

Algorithm B:

if $i > 1$ **then** $d = 2^{i-1}$; **else** $d = 2$;
for $j = 2^{i+1} \times \lfloor l/2^i \rfloor$ **to** $2^{i+1} \times \lfloor l/2^i \rfloor + d - 1$ **do**
if $i > 1$ **then** key input link pairs are $(j, j+d)$ and $(j+2^i, j+2^i+d)$; **else** key input link pairs are $(j, j+d)$;
end for;

Two common transitive output terminals for obtained key input link pairs are

when $l \bmod 2^i \neq 0$,
$n/2$ for input links j and $j+2^i$,
$n/2+2d$ for input links $j+d$ and $j+2^i+d$,
when $l \bmod 2^i = 0$,
$n/2+2$ for input links j and $j+2^i$,
$n/2+2+2d$ for input links $j+d$ and $j+2^i+d$.

Figure 3 shows the upper case, that is the case when $l \bmod 2^i \neq 0$. *Algorithm B* is applicable to the case of $n > 8$, see Appendix in the special case $n = 8$.

Example 2 *Let us consider a* 16×16 *banyan network with faulty switching element* E_{11}. *Input links 0, 1, 2, and 3 have no path to output links 2, 3, 6, 7, 10, 11, 14, 15 because of* E_{11}. *The key input link pairs are* $(0,2)$ *and* $(1,3)$ *because of* $i = 1$ *and* $d = 2$. *These four input links can be connected to the output links by allowing two cycles for each path between input and output links. Let us sum up and classify the required two passes by common required permutation products, that is,* $0 \to 8 \to 2$ *and* $2 \to 12 \to 10$ *(which require* Γ_2 *and* Γ_9*),* $0 \to 8 \to 3$ *and* $2 \to 12 \to 11$ *(*Γ_2 *and* Γ_8*),* $\ldots$, $0 \to 8 \to 15$ *and* $2 \to 12 \to 7$ *(*Γ_2 *and* Γ_{14}*),* $1 \to 8 \to 2$ *and* $3 \to 12 \to 10$ *(*Γ_{10} *and* Γ_9*),* $1 \to 8 \to 3$ *and* $3 \to 12 \to 11$ *(*Γ_{10} *and* Γ_8*),* $\ldots$, $1 \to 8 \to 15$ *and* $3 \to 12 \to 7$ *(*Γ_{10} *and* Γ_{14}*). We prepared* 2×16 *cycles. So, total* 3×16 *cycles are required.*

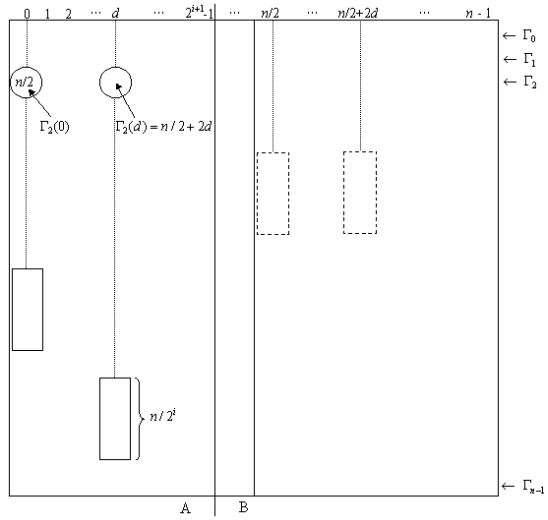


Fig. 3 An $n \times n$ matrix L' in the case of faulty switching element E_{il}.

We have considered the faulty switching element E_{il} $(0 < i \leq \log n - 2, 0 \leq l \leq n/4 - 1)$. Matrix L' in E_{il} $(0 < i \leq \log n - 2, n/4 \leq l \leq n/2 - 1)$ can be easily imagined as domains A and B are in apposition to Fig. 3, and necessary fault-tolerant routes can be constructed in the same manner. Though *Algorithm B* is available, two common transitive output terminals for obtained key input link pairs must be changed as follows:

When $l \bmod 2^i \neq 0$,
0 for input links j and $j+2^i$,
$2d$ for input links $j+d$ and $j+2^i+d$.
When $l \bmod 2^i = 0$,
2 for input links j and $j+2^i$,
$2+2d$ for input links $j+d$ and $j+2^i+d$.

Now, let us generalize our discussion. Since the number of input links which have a disconnected path to the output link is 2^{i+1}, then the total number of key input link pairs is $2^{i+1}/2 = 2^i$. The number of output links which have some disconnected paths from an input is

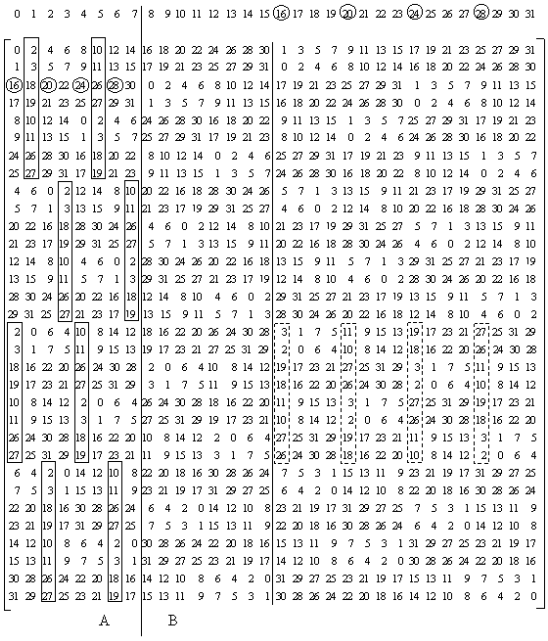


Fig. 4 Matrix L' in the case of 32×32 banyan network with faulty switching element E_{21}.

$n/2^i$. Therefore, the number of required extra cycles to construct fault-tolerant routes is $2^i \times (n/2^i) \times 2 = 2n$. From the above discussion, we could obtain a result that the upper bound of the number of required cycles to perform an all-to-all personalized communication in an $n \times n$ banyan network with a single internal fault is $3n$.

Fortunately, $n \times n$ banyan networks can have a lower upper bound than the $3n$ we just obtained, in the case of $n > 16$ where 3 or more cycles can be performed by a single permutation product. Next, we will discuss these cases. **Figure 4** shows L' in the case of 32×32 banyan network with faulty switching element E_{21}. Four routes $0 \to 16 \to 3$, $2 \to 20 \to 11$, $4 \to 24 \to 19$, and $6 \to 28 \to 27$ can be constructed by only 2 permutation products Γ_2 and Γ_{16}. Another four routes $0 \to 16 \to 2$, $2 \to 20 \to 10$, $4 \to 24 \to 18$, and $6 \to 28 \to 26$ can be constructed by Γ_2 and Γ_{17}, ...etc. By using these bypasses, all routes from each of the input links 0, 2, 4, and 6 to every output link can be constructed. By using pairs of permutation products Γ_{18} and Γ_{16}, Γ_{18} and Γ_{17}, Γ_{18} and Γ_{18}, ...etc., all routes from each of the input links 1, 3, 5, and 7 to every output link can be constructed. The total number of extra cycles is 8 (= the length of each square: $n/2^i) \times 2 \times 2 = n$ in this case. In this case we conclude n extra cycles and then the total $2n$ cycles are required to perform an all-to-all personalized communication.

Let us generalize this case. In matrix L' in the case of $n \times n$ banyan network with the faulty switching element E_{il}, domain A is an $n \times 2^{i+1}$ matrix. We will pay attention to "the number of entries which are in a row and in domain A but not in solid squares, and whose names are the same as columns which have entries in squares written by dotted lines in the same row and in domain B". It is obvious that this number is the number of routes between input and output links realized by two permutation products. In the above example, since a set of entries which are in the 3-rd row in domain A, but not in solid sequences, and whose names are the same as columns which have entries in squares written by dotted lines in the 17-th row and in domain B is $\{16, 20, 24, 28\}$, then the number is 4. Let us prove that this number is 2 when $i = 1$ or $\log n - 2$, otherwise 4.

From the property of Matrix L' for E_{il}, it is obvious that there always exists a row whose some entries in domain A are given by the following set X;

$$X = \{n/2, n/2+2, n/2+4, \ldots, n/2+2\cdot(2^{i+1}-1)\} \quad \text{when } i < \log n - 2, \text{ or}$$

$$X = \{n/2, n/2+2, n/2+4, \ldots, n/2+2\cdot(2^{\log n-2}-1)\} \quad \text{when } i = \log n - 2.$$

Therefore, a set X' whose elements in solid squares are excluded from X is given as

$$X' = \{n/2, n/2+2, n/2+4, \ldots, n/2+2\cdot(l \bmod 2^i)-2, n/2+2\cdot(l \bmod 2^i), n/2+2\cdot(l \bmod 2^i)+2, n/2+2\cdot(l \bmod 2^i)+4, \ldots, n/2+2\cdot(l \bmod 2^i)+4\cdot 2^{i-1}-2, n/2+2\cdot(l \bmod 2^i)+4\cdot 2^{i-1}+2, n/2+2\cdot(l \bmod 2^i)+4\cdot 2^{i-1}+4, \ldots, n/2+2\cdot(2^{i+1}-1)\} \quad \text{when } i < \log n - 2, \text{ or}$$

$$X' = \{n/2, n/2+2, n/2+4, \ldots, n/2+2\cdot(l \bmod 2^{\log n-2})-2, n/2+2\cdot(l \bmod 2^{\log n-2})+2, n/2+2\cdot(l \bmod 2^{\log n-2})+4,$$

$$\ldots, n/2 + 2 \cdot (2^{\log n - 2} - 1)\}$$
$$\text{when } i = \log n - 2.$$

From the property of Matrix L', it is also obvious that there exists the following set Y of columns whose entries belong to squares written by dotted lines in the same row and in domain B:

$$Y = \{n/2, n/2 + 2 \cdot 2^{i-1}, n/2 + 4 \cdot 2^{i-1}, n/2 + 6 \cdot 2^{i-1}, \ldots, n/2 + (n/2 - 2 \cdot 2^{i-1})\}$$
$$\text{when } l \bmod 2^i \neq 0, \text{ or}$$
$$Y = \{n/2 + 2, n/2 + 2 + 2 \cdot 2^{i-1}, n/2 + 2 + 4 \cdot 2^{i-1}, n/2 + 2 + 6 \cdot 2^{i-1}, \ldots, n/2 + 2 + (n/2 - 2 \cdot 2^{i-1})\}$$
$$\text{when } l \bmod 2^i = 0.$$

The number which we would like to obtain is $|X' \cap Y|$.

When $l \bmod 2^i \neq 0$,

$$X' \cap Y = \begin{cases} \{n/2, n/2 + 4 \cdot 2^{i-1}\} \\ \qquad \text{for} \quad i = 1, \\ \{n/2, n/2 + 2 \cdot 2^{i-1}, \\ \ n/2 + 4 \cdot 2^{i-1}, n/2 + 6 \cdot 2^{i-1}\} \\ \qquad \text{for} \quad 1 < i < \log n - 2, \\ \{n/2, n/2 + 2 \cdot 2^{i-1}\} \\ \qquad \text{for} \quad i = \log n - 2, \end{cases}$$

when $l \bmod 2^i = 0$,

$$X' \cap Y = \begin{cases} \{n/2 + 2, n/2 + 2 + 4 \cdot 2^{i-1}\} \\ \qquad \text{for} \quad i = 1, \\ \{n/2 + 2, n/2 + 2 + 2 \cdot 2^{i-1}, \\ \ n/2 + 2 + 4 \cdot 2^{i-1}, \\ \ n/2 + 2 + 6 \cdot 2^{i-1}\} \\ \qquad \text{for} \quad 1 < i < \log n - 2, \\ \{n/2 + 2, n/2 + 2 + 2 \cdot 2^{i-1}\} \\ \qquad \text{for} \quad i = \log n - 2, \end{cases}$$

From the above discussion, we obtain

$$|X' \cap Y| = \begin{cases} 4 & \text{for} \quad 1 < i < \log n - 2, \\ 2 & \text{otherwise.} \end{cases}$$

From the above discussion, the additional number of cycles because of single fault E_{il} is

$$(n/2^i) \cdot 2 \cdot (2^{i+1}/t),$$

that is, $n/(2^{-2} \cdot t)$ is obtained. Where $t = |X' \cap Y|$, and 2^{i+1} is the number of columns of domain A. This result can be summed up in the following theorem.

Theorem 1 *All-to-all personalized communication in a* $\log n$*-stage banyan network with the single faulty switching element* E_{il} *on any internal* i*-th stage can be achieved in cycles of the following upper bound* $U(i)$*:*

if $i = 1$ *or* $\log n - 2$,
$U(i) = 3n$.

if $i = 2, \ldots, \log n - 3$,
$U(i) = 2n$.

proof: It is obvious from the above discussion. **Table 1** shows the upper bound given by Theorem 1.

Table 1 The upper bound of the number of required cycles.

	faulty switching element							
n	E_{1l}	E_{2l}	E_{3l}	E_{4l}	E_{5l}	E_{6l}	E_{7l}	E_{8l}
2^4	$3n$	$3n$						
2^5	$3n$	$2n$	$3n$					
2^6	$3n$	$2n$	$2n$	$3n$				
2^7	$3n$	$2n$	$2n$	$2n$	$3n$			
2^8	$3n$	$2n$	$2n$	$2n$	$2n$	$3n$		
2^9	$3n$	$2n$	$2n$	$2n$	$2n$	$2n$	$3n$	
2^{10}	$3n$	$2n$	$2n$	$2n$	$2n$	$2n$	$2n$	$3n$

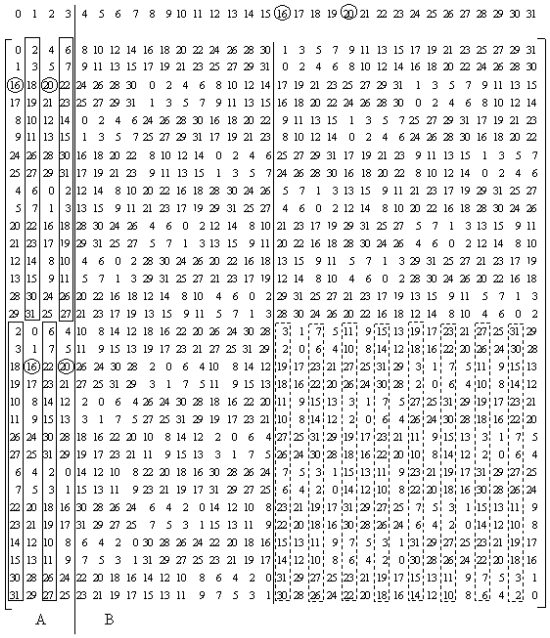


Fig. 5 Matrix L' in the case of 32×32 banyan network with faulty switching element E_{11}.

Example 3 *Let us consider a* 32×32 *banyan network with the faulty switching element* E_{11}*. Matrix* L' *in the case of* E_{11} *is shown in* **Fig. 5**. *Since set* X *is given by* $X = \{16, 18, 20, 22\}$*, set* X' *is given by* $X' = \{16, 20\}$ *and set* Y *is given by* $Y = \{16, 18, 20, 22, 24, 26, 28, 30\}$*. Therefore, it is* $X' \cap Y = \{16, 20\}$ *and* $|X' \cap Y| = 2$ *is obtained. This number* $t = 2$ *leads us to the additional number of cycles which is* $2n$*. Then, total* $3n$ *cycles are required to perform an all-to-all personalized communication in the banyan network.*

4. Conclusion

We have presented a method of fault-tolerant all-to-all personalized communication that can tolerate to a single non-critical fault. The proposed method has the upper bound of required cycles depending on the stages with faulty switching element. The results obtained in this paper say that a single fault on an internal stage compels the all-to-all personalized communication to pay double $\sim$ three times as much as the communication time is required in a nonfaulty case.

Acknowledgments The authors appreciate the useful comments from anonymous referee, which greatly improved the quality of this paper.

References

1) Johnsson, S.L. and Ho, C.T.: Optimum Broadcasting and Personalized Communication in Hypercubes, *IEEE Trans. Comput.*, Vol.C-38, No.9, pp.1249–1268 (1989).
2) Saad, Y. and Schultz, M.H.: Data Communication in Parallel Architectures, *Parallel Computing*, Vol.11, pp.131–150 (1989).
3) Scott, D.S.: Efficient All-to-All Communication Patterns in Hypercube and Mesh Topologies, *Proc. 6th Distributed Memory Computing Conference*, pp.398–403 (1991).
4) Thakur, R. and Choudhary, A.: All-to-All Communication on Meshes with Wormhole Routing, *Proc. 8th IEEE International Parallel Processing Symposium*, pp.561–565 (1994).
5) Yang, Y. and Wang, J.: Efficient All-to-All Broadcast in All-Port Mesh and Torus Networks, *Proc. 5th IEEE International Symposium on High-Performance Computer Architecture* (*HPCA-5*), Orlando, FL, pp.290–299 (1999).
6) Tseng, Y.-C. and Gupta, S.: All-to-All Personalized Communication in a Wormhole-Routed Torus, *IEEE Trans. Parallel and Distributed Systems*, Vol.7, No.5, pp.498–505 (1996).
7) Tseng, Y.-C., Lin, T.-H., Gupta, S. and Panda, D.K.: Bandwidth-Optimal Complete Exchange on Wormhole Routed 2D/3D Torus Network: A Diagonal-Propagation Approach, *IEEE Trans. Parallel and Distributed Systems*, Vol.8, No.4, pp.380–396 (1997).
8) Petrini, F.: Total-Exchange on Wormhole k-ary n-cubes with Adaptive Routing, *Proc. First Merged IEEE International Parallel Processing Symposium & Symposium on Parallel and Distributed Processing*, Orlando, FL, pp.267–271 (1998).
9) Suh, Y.J. and Yalmanchili, S.: All-to-All Communication with Minimum Start-up Costs in 2D/3D Tori and Meshes, *IEEE Trans. Parallel and Distributed Systems*, Vol.9, No.5, pp.442–458 (1998).
10) Suh, Y.J. and Shin, K.G.: Efficient All-to-All Personalized Exchange in Multidimensional Torus Networks, *Proc.1998 International Conference on Parallel Processing*, pp.468–475 (1998).
11) Yang, Y. and Wang, J.: All-to-All Personalized Exchange in Banyan Networks, *Proc. IASTED International Conference on Parallel and Distributed Computing and Systems*, Cambridge, MS, pp.78–86 (1999).
12) Park, S. and Bose, B.: All-to-All Broadcasting in Faulty Hypercubes, *IEEE Trans. Comput.*, Vol.46, No.7, pp.749–755 (1997).
13) Varma, A. and Raghavendra, C.S.: Fault-tolerant Routing in Multistage Interconnection Networks, *IEEE Trans. Comput.*, Vol.38, No.3, pp.385–393 (1989).

Appendix

Case $n = 8$ is a special case where for faulty switching element E_{11}, an example of extra permutation products sequence is as following $(2n+1)$-permutation-products sequence:

$$
\begin{array}{cc}
\begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{matrix} & \\
\begin{bmatrix}
0 & 2 & 4 & 6 & 1 & 3 & 5 & 7 \\
2 & 0 & 6 & 4 & 3 & 1 & 7 & 5 \\
7 & 5 & 3 & 1 & 6 & 4 & 2 & 0 \\
5 & 7 & 1 & 3 & 4 & 6 & 0 & 2 \\
1 & 3 & 5 & 7 & 0 & 2 & 4 & 6 \\
4 & 6 & 0 & 2 & 5 & 7 & 1 & 3 \\
6 & 4 & 2 & 0 & 7 & 5 & 3 & 1 \\
3 & 1 & 7 & 5 & 2 & 0 & 6 & 4 \\
0 & 2 & 4 & 6 & 1 & 3 & 5 & 7 \\
7 & 5 & 3 & 1 & 6 & 4 & 2 & 0 \\
4 & 6 & 0 & 2 & 5 & 7 & 1 & 3 \\
2 & 0 & 6 & 4 & 3 & 1 & 7 & 5 \\
3 & 1 & 7 & 5 & 2 & 0 & 6 & 4 \\
4 & 6 & 0 & 2 & 5 & 7 & 1 & 3 \\
7 & 5 & 3 & 1 & 6 & 4 & 2 & 0 \\
0 & 2 & 4 & 6 & 1 & 3 & 5 & 7 \\
3 & 1 & 7 & 5 & 2 & 0 & 6 & 4
\end{bmatrix} &
\begin{matrix}
\Gamma_0 \\ \Gamma_4 \\ \Gamma_7 \\ \Gamma_3 \\ \Gamma_1 \\ \Gamma_2 \\ \Gamma_6 \\ \Gamma_5 \\ \Gamma_0 \\ \Gamma_7 \\ \Gamma_2 \\ \Gamma_4 \\ \Gamma_5 \\ \Gamma_2 \\ \Gamma_7 \\ \Gamma_0 \\ \Gamma_5
\end{matrix}
\end{array}
$$

(Received December 8, 2000)
(Accepted July 2, 2001)

Masashi Yaku was born in 1978. He graduated from the Department of Information and Knowledge Engineering, Faculty of Engineering, Tottori University in 2000. He is now a graduate student of Tottori university. He is making a study of ATM switching networks.

Hiroshi Masuyama was born in 1943. He is now a professor in the Department of Information and Knowledge Engineering of Tottori University. He has been a professor in Miyazaki and Osaka Universities. He was also a visiting professor at Stanford University between 1990 and 1991, and Boston University in 1996. He has published papers on topics including fault tolerance of logical circuits, analysis and synthesis of parallel and distributed systems, and network algorithm.