Broadcasting in Faulty Binary Jumping Networks[1]

Yijie Han*, Yoshihide Igarashi**,[2], Kumiko Kanai*** and Kinya Miura**

$*$ Department of Computer Science, University of Hong Kong, Hong Kong

$**$ Department of Computer Science, Gunma University, Kiryu, 376 Japan

$***$ Research and Development Center, Toshiba Corporation, Kawasaki, 210 Japan

Abstract

We discuss the fault tolerance of an information disseminating scheme in a processor network called a binary jumping network. The following results are shown. Let N be the number of processors in the network. When N is a power of 2, $\log_2 N + f + 1$ rounds suffice for broadcasting in the binary jumping network if at most f processors and/or links have failed and $f \leq \log_2 N - 1$. For an arbitrary N, $\lceil \log_2 N \rceil + f + 2$ rounds suffice for broadcasting in the binary jumping network if at most f processors and/or links have failed and $f \leq \lceil \log_2 N \rceil - 2$. For an arbitrary N and $f = \lceil \log_2 N \rceil - 1$, $3\lceil \log_2 N \rceil$ rounds suffice for sending a message to a destination with a certain distance from the source processor. This result implies that for some N and arbitrary $f = \lceil \log_2 N \rceil - 1$, $3\lceil \log_2 N \rceil$ rounds suffice for broadcasting. For other N and arbitrary $f = \lceil \log_2 N \rceil - 1$, the bound given in this paper is larger than $3\lceil \log_2 N \rceil$.

1. Introduction

Data broadcasting in a network is a very fundamental operation for a distributed system. It can be accomplished by the data disseminating process in a network in a way that each processor repeatedly receives and forwards messages. We consider a scheme such that any processor can be the source of information and any round can be the start of broadcasting. This type of broadcasting schemes have been studied in [1, 4, 6, 10, 11]. They are somewhat different from broadcasting schemes with the fixed starting round in [2, 3, 5, 7, 9, 12-14], since the necessary number of rounds for broadcasting in a faulty network depends on the starting round.

In the network, multiple copies of a message are disseminated through disjoint paths, and the fault tolerance can be achieved by this multiplicity. The processors in the network are synchronized with a global clock. In the network there is a link from processor u to

[1] A preliminary version of this paper was presented at the 3rd International Symposium on Algorithms and Computation, ISSAC'92, Nagoya, Japan, December 16-18, 1992. See also *Lecture Notes Computer Science* Vol. 650, pp. 145-154. Springer-Verlag, New York/Berlin, 1992.

[2] E-mail igarashi@cs.gunma-u.ac.jp.

processor v if and only if $v-u$ modulo N is 2^k for some $0 \leq k \leq \lceil \log_2 N \rceil - 1$. This network is called the binary jumping network. The total number of links in the network is $N\lceil \log_2 N \rceil$. Information disseminating schemes that require only $\Theta(N)$ links are known [2, 8], but they are not time optimal. The scheme discussed in this paper can be traced back to the work by Knödel [11] and by Alon *et al.*[1]. Han and Finkel discussed the fault tolerance of the scheme in the case where at most one fault exists [6].

We consider only the case where faulty processors cannot forward messages, but can receive messages, and/or some links may be disconnected. We do not consider cases where a faulty processor alters information. The period for forwarding a message from a processor to one of its neighbors is called a *round*. We assume that each processor can forward a message and can receive a message in the same round. This assumption is different from the standard telephone model [7], but it is also reasonable since the time duration of each round may be long enough for sending a message and receiving a message in the same round. We also assume that the source processor is always faultless. We show that for an arbitrary N, $\lceil \log_2 N \rceil + f + 2$ rounds suffice for broadcasting if f faults exist and $f \leq \lceil \log_2 N \rceil - 2$. The case where $f = \lceil \log_2 N \rceil - 1$ is also discussed.

2. An Information Disseminating Scheme

Our motivations why we study binary jumping networks are as follows. First of all, for any N there exists a binary jumping network with N nodes. Second, binary jumping networks have regular connections like hypercubes, and thirdly there exists a time optimal scheme for broadcasting in binary jumping networks in absence of faults.

Definition 1. A directed graph $G = (V, E)$ is called a binary jumping network with N nodes (N processors) if $V = \{0, \cdots, N-1\}$ and $E = \{(u, v) | u, v \text{ in } V, \text{ and } v-u \text{ modulo } N \text{ is } 2^k \text{ for some } 0 \leq k \leq \lceil \log_2 N \rceil - 1\}$.

The number of processors in the network is denoted by N, and $\lceil \log_2 N \rceil$ is denoted by n. The notation $[m]_r$ means m modulo r. Throughout this paper, we consider the following scheme. When N is a power of 2, the scheme is the same one as in [1]. The correctness of the scheme in absence of faults can be derived from a classical result in [11].

```
procedure disseminate(N)
    repeat
        for round := 0 to ⌈log_2 N⌉ − 1 do
            for each processor u send a message
            from u to processor [u + 2^round]_N concurrently
    forever
```

Theorem 1 (Han and Finkel [6]). *Procedure disseminate will broadcast information from any source to all destinations within any consecutive*

(1) $\lceil \log_2 N \rceil$ *rounds if no processors have failed,*

(2) $\log_2 N + 1$ *rounds if N is a power of 2 and exactly one processor has failed, and*

(3) $\lceil \log_2 N \rceil + 2$ *rounds if exactly one processor or exactly one link has failed.*

3. Message Route Representation

In this section, we introduce message route representation to express multiple processor disjoint routes for sending messages by procedure *disseminate*.

Definition 2. Message route $u(i : a_r \cdots a_0)$ is a sequence of integers $(\alpha_0, ..., \alpha_{r+1})$ defined as $u(i : a_r \cdots a_0) = (\alpha_0, ..., \alpha_{r+1})$, where $(a_r \cdots a_0)$ is a binary sequence, and for each $0 \le j \le r+1$, $\alpha_j = u + \sum_{k=0}^{j-1}(a_k 2^{[i+k]_n})$. Each α_j denotes processor $[\alpha_j]_N$, and α_j is called a relabelled address. (Note that $\alpha_0 = u$ and that a relabeled address is used to distinguish an integer from the residual class it belongs to. If $\alpha_j < N$ then relabelled address α_j coincides with address $[\alpha_j]_N$.)

If a_t is the first nonzero from the rightmost of $a_r \cdots a_0$, $u(i : a_r \cdots a_0)$ is a message route by procedure *disseminate* from processor u such that the first message move is at round $[i+t]_n$. For all nonzero bits $a_{j_k}, ..., a_{j_0}$ ($j_{g-1} < j_g$ for all $1 \le g \le k$) in $a_r, \cdots, a_0$, sequence $(\alpha_0, \alpha_{j_0+1}, ..., \alpha_{j_k+1})$ denotes the ordered set of processors in the message route. It is also clear that the message flow through the message route $u(i; a_r \cdots a_0)$ takes $r+1$ rounds if a_r is nonzero.

Definition 3. Let u and v ($0 \le u, v \le N-1$) be a pair of processors. Then for each $0 \le i \le n-1$, $R_i(u, v) = u(i; a_n \cdots a_0)$, where $a_0 = 1$, and for each $k > 0$, $a_k = q_{[i+k]_n}$ if $q_{n-1} \cdots q_0$ is the binary representation of $[v - u - 2^i]_N$.

Note that the last processor of message route $R_i(u, v)$ is $[u + \sum_{k=0}^{n}(a_k 2^{[i+k]_n})]_N = v$. The message initially located in processor u does not move in message route $R_i(u, v)$ before round i. The next lemma is immediate.

Lemma 1 *For any $s, i (0 \le s, i \le n-1)$ and any pair of processors u and v, the message from processor u by procedure disseminate with starting round s will reach processor v through message route $R_{[s+i]_n}(u, v)$ within at most $n+i+1$ rounds if no processors and no links in the route have failed.*

4. Fault Tolerance When $N = 2^n$

Because of the symmetry of the network, without loss of generality we may assume that in the proofs of the following lemma and theorem the source processor is 0.

Lemma 2 *When N is a power of 2, for any $i_1, i_2 (0 \le i_1 < i_2 \le n-1)$, u and v ($0 \le u, v \le N-1$), the set of processors in $R_{i_1}(u, v)$ and the set of processors in $R_{i_2}(u, v)$ are disjoint except for the source and the destination.*

Proof. We may assume that each processor is addressed as a binary number and that the source is processor 0. Suppose that $R_{i_1}(0, v) = (\alpha_0, ..., \alpha_{n+1})$ and $R_{i_2}(0, v) = (\beta_0, ..., \beta_{n+1})$. The (i_1+1)st bit from the rightmost of the binary representation of any processor in $(\alpha_0, ..., \alpha_{n+1})$ except for processor 0 and v is 1, and the (i_2+1)th bit from the rightmost of the binary representation of any processor in $(\beta_0, ..., \beta_{n+1})$ except for processors 0 and v is 1. Let the binary representation of v be $v_{n-1} \cdots v_0$.

(I) Case where $v_{i_1} = 0$. The $(i_1 + 1)$th bit from the rightmost of the binary representation of any processor in $(\beta_0, ..., \beta_{n+1})$ is 0, whereras the correponding bit of the binary representation of any processor in $(\alpha_0, ..., \alpha_{n+1})$ except for processor 0 and v is 1. Hence, the set of processors in $(\alpha_0, ..., \alpha_{n+1})$ and the set of processors in $(\beta_0, ..., \beta_{n+1})$ are disjoint except for processors 0 and v.

(II) Case where $v_{i_1} = 1$ and $v_{i_2} = 0$. The proof is similar to the previous case.

(III) Case where $v_{i_1} = 1$ and $v_{i_2} = 1$. The $(i_1 + 1)$th bit from the rightmost of the binary representation of any processor in $(\beta_0, ..., \beta_{n+i_1-i_2})$ is 0, whereas the corresponding bit of the binary representation of any processor in $(\alpha_1, ..., \alpha_{n+1})$ is 1. Hence, the set of processors in $(\alpha_1, ..., \alpha_{n+1})$ and the set of processors in $(\beta_0, ..., \beta_{n+i_1-i_2})$ are disjoint except for processor 0. The $(i_2 + 1)$th bit from the rightmost of the binary representation of any processor in $(\alpha_0, ..., \alpha_{i_2-i_1})$ is 0, whereas the corresponding bit of the binary representation of any processor in $(\beta_1, ..., \beta_{n+1})$ is 1. Hence, the set of processors in $(\alpha_0, ..., \alpha_{i_2-i_1})$ and the set of processors in $(\beta_0, ..., \beta_{n+1})$ are disjoint except for processor 0.

Assume that w is a commom processor in $(\alpha_{i_2-i_1+1}, ..., \alpha_{n+1})$ and $(\beta_{n+i_1-i_2+1}, ..., \beta_{n+1})$. Let the binary representation of w be $w_{n-1} \cdots w_0$. Then $w_k = v_k$ for $i_1 \leq k < i_2$ because w is in $(\alpha_{i_2-i_1+1}, ..., \alpha_{n+1})$, and $w_k = v_k$ for $k < i_1$ or $k \geq i_2$ because w is in $(\beta_{n+i_1-i_2+1}, ..., \beta_{n+1})$. Therefore, the set of processors in $(\alpha_0, ..., \alpha_{n+1})$ and the set of processors in $(\beta_0, ..., \beta_{n+1})$ are disjoint except for processors 0 and v. □

Theorem 2 *Procedure disseminate will broadcast from any source to all destinations within any consecutive* $\log_2 N + f + 1$ *rounds if* N *is a power of* 2 *and at most* $f \leq \log_2 N - 1$ *processors and/or links have failed.*

Proof. Assume that the source is processor 0. Let s be the starting round and f be the number of faulties. Suppose that $f \leq \log_2 N - 1$. Consider the message routes $R_{[s+i]_n}(0, v)$ $(i = 0, \cdots, f)$. From Lemma 2, these $f + 1$ message routes are processor disjoint except for the source and the destination. Since we assume that the source is always faultless, at least one of these message routes is faultless. Hence, from Lemma 1 the message will reach processor v within at most $\log_2 N + f + 1$ rounds. □

For a broadcasting scheme on a network with N nodes, if it happens that all the first f trials of sending the message fail, then $\log_2 N + f$ rounds are not sufficient to complete the broadcasting. Therefore, no scheme on the network cannot tolerate and broadcast in less than $n + f + 1$ rounds. Thus the bound given in Theorem 2 is tight.

5. Fault Tolerance for an Arbitrary N

Assume that the source of information is processor 0. We divide the message routes $R_i(0, v)$, $0 \leq i \leq n - 1$ into two classes, *ascending routes* and *nonascending routes.*

Definition 4. A message route $R_i(0.v)$ is called an ascending route if $2^i \leq v$, and otherwise it is called a nonascending route.

The next lemma is immediate from the above definition.

Lemma 3 *The sequence of processors in an ascending route is in ascending order.*

Due to Lemma 3, we need to consider only the processors $0, 1, 2, \cdots, v$ for ascending routes. Therefore, when we consider only ascending routes, the sequence of processors in $R_i(0, v)$ with N processors and that in $R_i(0, v)$ with $2^{\lceil \log_2 N \rceil}$ processors are the same. Hence, from the argument in the proof of Lemma 2 the next lemma is immediate.

Lemma 4 *For an arbitrary size of the network all ascending routes are processor disjoint except for the source and the destination.*

We now consider nonascending routes $R_i(0, v)$, where relabeled addresses are used. The relabeled address of processor p is p itself or $p + N$. The sequence of relabeled addresses of the processors in $R_i(0, v)$ is in ascending order and ends with $v + N$. Hence, any processor except for processor 0 can appear at most once in the route. For example, the sequence of processors in $R_3(0, 6)$ with $N = 9$ is $(0, 8, 0, 2, 6)$ and it is expressed as $(0, 8, 9, 11, 15)$ in relabeled addresses. The sequence of relabeled addresses of the processors in nonascending route $R_i(0, v)$ with network size N is exactly the same as the sequence of processors in $R_i(0, v + N)$ with network size 2^n or 2^{n+1}. Therefore, following the proof of Lemma 2, all the nonascending routes are processor disjoint except for the source and destination. Note that only one of the nonascending routes may go through the source processor. Since we assume that the source processor is not faulty, we have the next lemma.

Lemma 5 *For an arbitrary size of the network, if t nonascending routes are available then they can be used for tolerating $t - 1$ faults.*

We now discuss the interaction between ascending routes and nonascending routes. Let destination v satisfy $2^{k-1} \leq v < 2^k < N$. Then for any nonascending route, each of rounds $0, 1, ..., k - 1$ will be executed only once. (Note that round k means the round when communication is performed from u to $[u + 2^k]_N$ for each processor u.)

Lemma 6 *Let $2^{k-1} \leq v < 2^k < N$. If nonascending route $R_i(0, v)$ did not reach processor v at the end of round $k - 1$, then it will not go through any processor in $\{0, ..., v\}$ during the rest of the rounds.*

Proof. Consider a nonascending route satisfying the condition of the lemma. Since the route must reach $v + N$, the message will be sent from a processor p to the processor v (its relabeled address is $v + N$) during the rest of the rounds $k, ..., i$. Hence, $p \leq v + N - 2^k < N$. Therefore, the nonascending route never goes through any processor in $\{0, ..., v\}$ (their relabeled addresses are in $\{N, ..., v + N\}$. $\square$.

Lemma 7 *At most one nonascending route may share an intermediate processor (i.e., a processor that is neither 0 nor v) with at most one ascending route.*

Proof. Let $2^{k-1} \leq v < 2^k$, and let S be the set of the relabeled addresses of all the nodes of nonascending routes immediately after the first round $n-1$. Let p_i be the ith largest relabeled address in S. Since every member of S is a multiple of 2^k, for every $i \geq 2$, $p_i \leq v + N - 2^k$. Hence, for every $i \geq 2$, the nonascending route that goes through p_i does not reach the destination before the end of round $k-1$. From Lemma 6, any nonascending route that goes through p_i ($i \geq 2$) never goes through any processor in $\{0, ..., v\}$. On the other hand, ascending routes go through only processors in $\{0, ..., v\}$. The nonascending route that goes through p_1 may share an intermediate processor with an ascending route. If the nonascending route first meets a processor of an ascending route, then the rest of the nonascending route overlaps with the rest of the ascending route. Hence, the nonascending route does not share any intermediate processor with other ascending routes. $\square$

From Lemmas 4, 5, and 7 we have the next theorem.

Theorem 3 *Procedure disseminate will broadcast from any source to all destinations within any consecutive* $\lceil \log_2 N \rceil + f + 2$ *rounds if at most* $f \leq \lceil \log_2 N \rceil - 2$ *processors and/or links have failed.*

Example 1. Consider the massage routes $R_i(0,6)$ $(0 \leq i \leq 5)$ with $N = 37$. These message routes are shown in Table I. Routes $R_3(0,6), R_4(0,6)$ and $R_5(0,6)$ are nonascending, whereas routes $R_0(0,6), R_1(0,6)$ and $R_2(0,6)$ are ascending. Let S be the set of the relabeled addresses in the nonascending routes immediately after the first round 5. Then $S = \{16, 32, 40\}$. The nonascending route with the maximum element in S could possibly go through a processor in an ascending route. In fact, processor 4 is included in both $R_2(0,6)$ and $R_3(0,6)$. If processors 1, 2, 4, and 32 have failed, only route $R_4(0,6)$ is not faulty among the routes $R_i(0,6)$ $(0 \leq i \leq 5)$. In this case, if the starting round is 5 then 12 rounds are necessary and sufficient for broadcasting. This number of rounds coincides with the bound given in Theorem 3.

6. The Case of $\lceil \log_2 N \rceil - 1$ Faults

We know that the vertex connectivity of the binary jumping network with N processors is $\lceil \log_2 N \rceil$. This fact can be derived from the result by van Doorn [15]. Hence, procedure *disseminate* should tolerate up to $\lceil \log_2 N \rceil - 1$ faults. However, van Doorn did not discuss in [15] how to construct disjoint paths connecting each pair of nodes. In this section we discuss how many rounds suffice to tolerate $\lceil \log_2 N \rceil - 1$ faults. Assume that the source is processor 0 and that v is a destination processor.

Theorem 4 *Let* $N = 2^{n-1} + t$ *and* $2^{k-1} \leq v < 2^k$. *If* $t + v \geq 2^k$ *and the number of faulty processors and/or links* $f \leq \lceil \log_2 N \rceil - 1$, *then* $3\lceil \log_2 N \rceil - k - 1$ *rounds suffice for sending a message from processor* 0 *to processor* v.

Proof. Let $t + v \geq 2^k$. Let $R = R_r(0, v)$ be the nonascending route that reaches the processor with the largest relabeled address p at the end of the first round $n-1$. As shown in the proof of Lemma 7, only route R could possibly intersect with an ascending route. It is obvious that p must be a multiple of 2^k.

Table I
Message Routes $R_i(0,6)$ $(0 \leq i \leq 5)$ with $N = 37$

		Round $\longrightarrow$											
$R_i(0,v)$		0	1	2	3	4	5	0	1	2	3	4	5
$R_0(0,6)$	0	1	-	5	-	-	-	6					
ascending													
$R_1(0,6)$	0	-	2	6									
ascending													
$R_2(0,6)$	0	-	-	4	-	-	-	-	6				
ascending													
$R_3(0,6)$	0	-	-	-	8	-	3	4	6				
nonascending							(40)	(41)	(43)				
$R_4(0,6)$	0	-	-	-	-	16	-	17	19	-	27	6	
nonascending												(43)	
$R_5(0,6)$	0	-	-	-	-	-	32	33	35	-	6		
nonascending											(43)		

Note. (r) indicates that r is a relabeled address.

(I) $N + v - p \geq 2^k$. Since $p \leq N + v - 2^k$, R cannot reach $N + v$ before the end of the first round $k - 1$. Hence, from Lemma 6, R cannot intersect with any ascending route.
(II) $N + v - p < 2^k$. Since we assume $2^k \leq t + v$ and $N + v - p < 2^k$, $p > 2^{n-1}$. In route R, at the first round $n - 1$ the message in processor $p' = p - 2^{n-1}$ moves to processor p. We now modify the route R and obtain route R'. In route R', the message in processor p' will not move at the first round $n - 1$. We now use rounds $0, ..., k - 1$ following the first round $n - 1$ to send the message in processor p' to processor $p'' = N + v - 2^{n-1}$. This move is possible since $N + v - p < 2^k$. We then use the second round $n - 1$ to move the message in processor p'' to the destination v (Note that $p'' + 2^{n-1} = N + v$). In the duration of moving the message from processor p' to processor p'', the route R' does not intersect with any of routes $R_i(0,v)$ except for route $R = R_r(0,v)$. At the second round $n - 1$ the message in processor p'' can reach directly the destination v. By replacing R with R', we obtain a set of n processor disjoint routes connecting processor 0 and processor v. Therefore, the network can tolerate $n - 1$ faults for sending the message from the source to processor v.

For any starting round, $2n$ rounds suffice for sending the message through the routes except for the modified route R'. When the starting round is round $k + 1$, the number of rounds required to send the message through R' is the largest. In the worst case, $3n - k - 1$ rounds may be required to send the message through R'. $\square$
Example 2. Let $N = 37 = 32 + 5$, $t = 5$, and let the source and the destination be processors 0 and $v = 6$, respectivley. Then $2^2 \leq v < 2^3$ and $k = 3$. The condition in Theorem 4, $t + v = 11 > 2^3$ is satisfied. The message routes $R_i(0,6)$ $(0 \leq i \leq 5)$ are shown in Table I. Only route $R_3(0,6)$ intersects with an ascending route $R_2(0,6)$. Route $R' = (0, 8, 9, 11, 43)$ (43 modulo 37 = 6) is obtained by modifying $R_3(0,6)$. This route is

shown in Table II. Routes $R_i(0,6)$ $(i = 0,1,2,4,5)$ and R' are processor disjoint message routes from processor 0 to processor 6. If the starting round is 4, then 14 rounds are required to send the message from processor 0 to processor 6 through route R'.

Table II
The modified Route R' in the Network with $N = 37$

		Round $\longrightarrow$											
		0	1	2	3	4	5	0	1	2	3	4	5
R'	0	-	-	-	8	-	-	9	11	-	-	-	6
modified route													(43)

Note. (r) indicates that r is a relabeled address.

Theorem 5 *Let $N = 2^{n-1} + t$ $(t \neq 0)$, $2^{k-1} \leq v < 2^k$, $t + v < 2^k$, and $m = \lfloor (2^k - v)/t \rfloor$. Then the message from processor 0 can reach processor v by procedure disseminate within any consecutive $n(m+2)$ rounds if at most $f \leq \lceil \log_2 N \rceil - 1$ processors and/or links have failed.*

Proof. Let S be the set of relabeled addresses of nonascending routes immediately after the first round $n-1$ as defined in the proof of Lemma 7. Note that each member of S is a multiple of 2^k. Since $t + v < 2^k$, we have $N + v < 2^{n-1} + 2^k$. Hence. among the nonascending routes, $R_{n-1}(0,v)$ has the largest relabeled address in S. As shown in the proof of Lemma 7, only $R_{n-1}(0,v)$ could possibly intersect with an ascending route. Note that $R_{n-1}(0,v)$ first goes through processor 2^{n-1}. Since $v \geq 2^{k-1}$ and $t + v < 2^k$, we have $t < 2^{k-1}$ and then $m \geq 1$. We construct a route R from processor 0 to processor v as follows. The first $m+2$ processors in R are $0, 2^{n-1}, q_1, q_2, ..., q_m$, where $q_j = 2^k - jt$ $(1 \leq j \leq m)$. This construction is possible since $[2^k - jt + 2^{n-1}]_N = 2^k - (j+1)t$ $(0 \leq j \leq m-1)$. In the following rounds 0 to $k-1$ we can move the message in processor q_m to processor $v + t$, since $v + t - (2^k - mt) < t < 2^{k-1}$. The last link in R is the edge from processor $v + t$ to processor v. This edge exists since $[v + t + 2^{n-1}]_N = v$. The route R never use processors from $\{1, ..., v-1\}$ nor processors from $\{2^k, ..., N-1\} - \{2^{n-1}\}$. Hence, route R is processor disjoint from all the ascending routes and the nonascending routes except for $R_{n-1}(0,v)$ that goes through processor 2^{n-1}. Thus $\lceil \log_2 N \rceil$ messages routes (i.e., all the ascending routes, all the nonascending routes except for $R_{n-1}(0,v)$, and route R) are processor disjoint. Route R requests the largest number of rounds among these $\lceil \log_2 N \rceil$ message routes. Note that for route R the worst choice of the starting round is round 0. The sequence of rounds used to move the message through route R is $n-1, k, m$ times $n-1$, some combination of rounds in $\{0, ..., k-1\}$, $n-1$. Hence, $n(m+2)$ rounds suffice for sending the message through route R to processor v. □

Corollary 1 *Let $N = 2^{n-1} + t$ and $t > 2^{n-3}$. Then procedure disseminate will broadcast from any source to all destinations within any consecutive $3\lceil \log_2 N \rceil$ rounds if at most $\lceil \log_2 N \rceil - 1$ processors and/or links have failed.*

Proof. If $v \geq 2^{n-1}$, then all $R_i(0,v)$ $(0 \leq i \leq n-1)$ are ascending routes and processor disjoint. Hence, in this case $2n$ rounds suffice for tolerating $f \leq \lceil \log_2 N \rceil - 1$ faults. Suppose that $2^{k-1} \leq v < 2^k$ and $k \leq n-1$. From Theorem 4 we may only consider destinations v such that $t + v < 2^{n-1}$. Since we assume that $t > 2^{n-3}$, $m = \lfloor (2^k - v)/t \rfloor = 1$. Then from Theorem 6, $3n$ rounds suffice for tolerating $f \leq \lceil \log_2 N \rceil - 1$ faults. □

7. Concluding Remarks

Using the message route representation $R_i(u,v)$ we have derived a sufficient number of rounds for broadcasting in a binary jumping network. From our computer experiment, for many values of N the bound given in Theorem 3 is tight. In Theorem 4, Theorem 5 and Corollary 1 we have partly solved the fault tolerance of procedure *disseminate* when $\lceil \log_2 N \rceil - 1$ faults exist. For an arbitrary N, a sufficient number of rounds to tolerate $\lceil \log_2 N \rceil - 1$ faults can be given from Theorem 4 and Theorem 5. The bound given in Theorem 5 is in general greater than $3\lceil \log_2 N \rceil$ rounds. At present we do not know whether for an arbitrary N and $f = \lceil \log_2 N \rceil - 1$, $3\lceil \log_2 N \rceil$ rounds suffice for broadcasting by procedure *disseminate*. Our computer experiment could not find any example of N processors including $f = \lceil \log_2 N \rceil - 1$ faulty processors $(N \leq 46)$, $\lceil \log_2 N \rceil - 1$ faulty links $(N \leq 20)$, or the sum of faulty processors and faulty links $= \lceil \log_2 N \rceil - 1$ $(N \leq 16)$ such that it requires more than $3\lceil \log_2 N \rceil$ rounds to complete broadcasting.

Acknowledgments

The authors thank Mr. Koji Obokata and Mr. Shingo Osawa for their assistance with our computer experiment, and the anonymous referees for their efforts and comments that helped in improving the presentation and readability of the paper.

References

1. N. Alon, A. Barak and U. Mauber, On disseminating information reliably without broadcasting, *The 7th International Conference on Distributed Computing Systems*, 1987, pp.74-81.

2. J-C. Bermond and C. Peyrat, Broadcasting in de Bruijn networks, *Congr. Numer.*, **66**, (1988) 283-292.

3. S.-C Chau and A. L. Liestman, Constructing fault-tolerant minimal broadcast networks. *J. Combin. Inform. System Sci.* **11**(1986), 1-18.

4. S. Carlsson, Y. Igarashi, K. Kanai A. Lingas, K. Miura and O. Petersson, Information disseminating schemes for fault tolerance in hypercubes, *IEICE Trans. on Fundam. Electron., Comm. Comput. Sci.* **E75-A** (1992) 255-260.

5. L. Gargano and U. Vaccaro, Minimum time broadcast networks tolerating a logarithmic number of faults. *SIAM J. Discrete Math.* **5** (1992), 178-198.

6. Y. Han and R. Finkel, An optimal scheme for disseminating information, *Proceedings*

of 22nd International Conference on Parallel Processing, St. Charles, Illinois, 1988, 198-203.

7. S. M. Hedetniemi, S. T. Hedetniemi, and A. L. Liestman, A survey of gossiping and broadcasting in communication networks. *NETWORKS* **18** (1988), 319-349.

8. M. Imase, T. Soneoka and K. Okada, Connectivity of regular directed graphs with small diameters, *IEEE Trans. Comput.* **34** (1985).267-273.

9. S. L. Johnsson and Ching-Tien Ho, Optimal broadcasting and personalized communication in hypercubes, *IEEE Trans. Comput.* **38**, (1989) 1249-1268.

10. K. Kanai, K. Miura and Y. Igarashi, Fault tolerance of Han-Finkel's scheme for disseminating information. Technical Report CS-91-3, Dept. of Computer Science, Gunma University, 1991.

11. W. Knödel, New gossiping and telephones, *Discrete Math.*, **13** (1975), 95.

12. A. L. Liestman, Fault-tolerant broadcast graphs, *NETWORKS* **15** (1985), 159-171.

13. D. Peleg and A. A. Schäffer, Time bounds on fault-tolerant broadcasting. *NETWORKS* **19** (1989), 803-822.

14. P. Ramanathan and K. G. Shin, Reliable broadcasting in hypercube multiprocessors, *IEEE Trans. Comput.* **37** (1988), 1654-1657.

15. E. A. van Doorn, Connectivity of circulant digraphs, *J. Graph Theory* **10** (1986), 9-14.

YIJIE HAN received the M. Sc. and Ph.D. from the Department of Computer Science, Duke University, in 1984 and in 1987, respectively. He took an assistant professorship in the Department of Computer Science at the University of Kentuky in 1987, and worked there until September 1992. He moved to the Department of Computer Science at the University of Hong Kong, and worked there as an associate professor until August 1993. Since September 1993 he has been working on programs and software design with various companies in Kentucky. His research interests include the design and analysis of computer algorithms and parallel and distributed computing.

YOSHIHIDE IGARASHI received the B.Sc. in electrical engineering in 1962, and the M.Sc. and Ph.D. in electrical and communication engineering in 1968 and in 1971, respectively, from Tohoku University. He was a research associate at Tohoku University in 1971-1974, a visiting fellow at the University of Edinburgh in 1972-1974, a lecturer at the University of Leeds in 1974-1977, and a lecturer at the City University in 1977-1978. In 1978 he joined Gunma University, where he is currently a professor of computer science. He was a visiting associate professor and a visiting professor at the University of Kentucky in

1980-1981 and in 1987-1988, respectively. His research interests include theory of computing and parallel algorithms.

KUMIKO KANAI received the B.Sc. in computer science and the M.Sc. in computer science from Gunma University in 1990 and in 1992, respectively. In April 1992 she joined the Research and Development Center of Toshiba Corporation, where she has been working on communication systems. Her research interests include communication systems, fault tolerant systems, and parallel and distributed computing.

KINYA MIURA received the B.Sc. in information science and the M.Sc. in information science from Kyoto University in 1983 and in 1985, respectively. He worked as a research associate in the Department of Computer Science at Gunma University from April 1989 to March 1994. In April 1994 he joined the Nara Advanced Institute of Science and Technology, where he has been working as a research associate. His research interests include higher order logic, information disseminating schemes, and computer algorithms.