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

A Fault-Tolerant Multicast Algorithm for Wormhole Routed Hypercubes

Shih-Chang Wang, Jeng-Ping Lin and Sy-Yen Kuo

Department of Electrical Engineering
National Taiwan University

Abstract

In this paper, we propose a novel fault-tolerant multicast algorithm for n-dimensional wormhole routed hypercubes. The multicast algorithm will remain functional if the number of faulty nodes in an n-dimensional hypercube is less than n. Our approach is not based on adding physical or virtual channels to the network topology. Instead, we integrate several techniques such as partitioning of nodes, partitioning of channels, node label assignments, and path-like multicast to achieve fault tolerance. Both theoretical analysis and simulation are performed to demonstrate the effectiveness of the proposed algorithm.

1.Introduction

In order to construct large-scale multicomputers, direct networks have become a popular interconnection architecture. The boolean n-dimensional hypercube(n-cube) computer is an interconnection with 2^n processors(nodes). Processors in an n-cube communicate by passing messages. The delivery of the same message from one source node to an arbitrary number of destination nodes is called the ***multicast*** communication. Efficient multicast communication has been shown to be useful in applications such as parallel simulation, as well as in operations such as replication and barrier synchronization[3].

A desirable multicast communication involves the following requirements[7,9]. First, the message transmission time from the source node to each of the destination nodes should be as short as possible. Second, the network traffic caused by a multicast communication should be as little as possible. Third, the algorithm for multicast communication must not be computationally complex. Finally, the multicast communication must be ***deadlock-free***[2,4]. Deadlock occurs when a packet waits for an event that cannot happen.

In this paper we consider hypercube structures that use ***wormhole routing*** rather than ***store-and-forward***, ***circuit switching***, or ***virtual cut-through*** switching techniques. Wormhole routing also uses a cut-through approach to switching. A packet consists of a sequence of ***flits*** (flow control digits)[2]. The size of a flit depends on system parameters, in particular the channel width. The header flit(s) of the packet governs the route, and the remaining flits of the message follow in a pipeline fashion (see Figure 1).

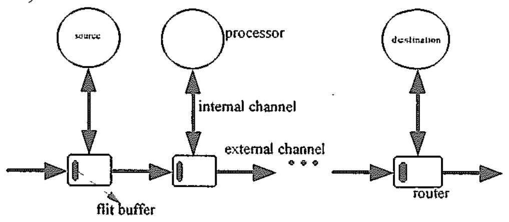

Figure 1. An illustration for wormhole routing.

Many deadlock-free routing(one-to-one)[8,11] and multicast(one-to-many)[5] algorithms for fault-free direct networks have been developed. One disadvantage of the wormhole routing compared with other switching techniques is that it tends to support routing which is less fault tolerant[15]. Under wormhole routing, each packet may reserve more than one channel at a time for its exclusive use. The order in which the channels of the network are reserved must be restricted to prevent deadlock among the packets. Such restrictions make routing less fault tolerant. Recently, some researchers[10] have attempted to use ***virtual channels***[1] to design routing algorithms that are fault tolerant in mesh structures. This paper examines the design of fault-tolerant deadlock-free multicast algorithms for wormhole-routed hypercubes.

The paper is organized as follows. In Section 2, we define the system model and describe the assumptions. The network partitioning strategies are described in Section 3, and a fault-tolerant deadlock-free multicast algorithm is shown in Section 4. The performance of the proposed algorithm is discussed in Section 5. Section 6 concludes this paper.

2. System Model and Assumptions

This section introduces the system model and assumptions that are commonly made in the literature[2,4,9,11]. We assume the following.

- The direct network we consider is the n-cube topology.
- A packet arriving at its destination node is eventually consumed.
- A node can generate multicast packets to an arbitrary number of destination nodes.

- Two neighboring nodes are connected by two opposite unidirectional channels.
- Each unidirectional channel has its own flit buffer.
- An unidirectional channel from node X to node Y is the outgoing channel of node X and the incoming channel of Y.
- Faults are static and detectable.
- The router for a node is considered to be part of the node. Both node faults and channel faults can be tolerated. If any of the incoming channels for a node is faulty, this node will be treated as faulty. Thus, only the node faults will be considered in this paper .
- A packet sent to a faulty node will be lost.
- A faulty node cannot generate packets and compute.
- Each non-faulty node allows packet replication and therefore, copies of a flit can be sent on multiple outgoing channels.
- There may exist an ***isolated*** non-faulty node if the number of faulty nodes is greater than n-1 in an n-cube. Therefore, we assume the number of faulty nodes is less than n.

3. Network Partitioning Strategies in Faulty Hypercubes

Fault-tolerant deadlock-free multicast communications are highly demanded for reliable and efficient parallel computations in multicomputers. In this section, we present network partitioning strategies, which are fundamental to the heuristic fault-tolerant deadlock-free multicast algorithm to be described in the next section.

3.1 Partitioning of Nodes

Several researchers have examined the partitioning scheme of nodes in n-cubes[11,12]. The partitioning scheme for nodes in an n-cube is attractive because it can make complex problems simple. Each node in an n-cube is identified by n coordinates, $(x_{n-1}, x_{n-2},..., x_0)$, where $x_i \in \{0,1\}$ for $0 \le i \le n-1$. The dimension associated with x_i is called the ith dimension. Two nodes $(x_{n-1}, x_{n-2},..., x_0)$ and $(y_{n-1}, y_{n-2},..., y_0)$ are said to be ***neighbors*** if $x_i=y_i$ for all i except one, j, where $x_j \neq y_j$. An ***m-partition*** of an n-cube is to partition the n-cube into 2^{n-m} m-cubes. An m-cube within an n-cube is denoted by $M=(x_{n-1}, x_{n-2},..., x_0)$, where exactly m of the x_i's are *'s(don't cares) and the remaining x_i's are either 0's or 1's. Let the set $P \subseteq \{0, 1,..., n-1\}$ and $|P|=2$. A 2-partition based on P for an n-cube contains 2^{n-2} 2-cubes where a 2-cube M is in the 2-partition if for all i, $0 \le i \le n-1$ and $i \in P$, $x_i=*$. The dimensions associated with x_i's, where $i \in P$, are called ***internal dimensions***; otherwise they are called ***external dimensions***. Two 2-cubes $(x_{n-1}, x_{n-2},..., x_0)$ and $(y_{n-1}, y_{n-2},..., y_0)$ in a 2-partition based on P for an n-cube are said to be ***adjacent*** if $x_i=y_i$ for all $i \notin P$ except one, j, where $x_j \neq y_j$. The neighboring nodes of a node W within the same 2-cube are called the ***buddies*** of node W. Let F be the set of faulty nodes in an n-cube. A 2-cube S is a ***fault-tolerant 2-cube*** if S contains at most one faulty node. A 2-partition based on P is a ***fault-tolerant 2-partition*** of an n-cube with a faulty set F if for every 2-cube S which is in the 2-partition, S is a fault-tolerant 2-cube. We will need the following lemma in [11] for the development of our partitioning scheme of nodes.

Lemma 1: For $n \ge 2$, given a set F of n-1 or fewer faulty nodes in an n-cube, there exists a fault-tolerant 2-partition of the n-cube. $\square$

For example, let F be the set {000000, 100001, 111000, 000100} in a 6-cube. If we let P be the set {0, 5}, faulty nodes 000000 and 100001 are in the same 2-cube *0000*. From the above definitions, the 2-partition with P={0, 5} is not a fault-tolerant 2-partition. However, if we let P={0, 1}, each 2-cube in the 2-partition based on P contains at most one faulty node and therefore, it is a fault-tolerant 2-partition. The algorithm to find a fault-tolerant 2-partition for ***Lemma 1*** is shown in Figure 2. This algorithm is derived directly from the proof of ***Lemma 1*** in [11]. It is straightforward to verify that the complexity of this algorithm is O(n). Therefore, given a fault-tolerant 2-partition of a faulty n-cube, we obtain ***Theorem 1***[12].

```
Algorithm: Find_A_Fault_Tolerant_2-Partition
/* Input: A faulty set $F$ with at most $n$-1 faulty nodes. */
/* Output: A fault-tolerant 2-partition with set $P$. */
Step 1: FOR $i$=0 TO $n$-1 DO
            Let $F_1$ be the set that deletes the $i$th dimension $x_i$ of each element in the set $F$;
            /* Note that each element in $F_1$ is a binary number with $n$-1 bits. */
                IF there is no repetitive element in $F_1$
                  THEN go to Step 2;
                END_IF
        END_FOR
Step 2: FOR $j$=0 TO $n$-2 DO
            Let $F_2$ be the set that deletes the $j$th dimension $x_j$ of each element in the set $F_1$;
                /* Note that each element in $F_2$ is a binary number with $n$-2 bits. */
                IF there is no repetitive element in $F_2$
                  THEN go to Step 3;
                END_IF
        END_FOR
Step 3: Let $P$ be the set {$i$, $j$}; return $P$;
```

Figure 2. The algorithm to find a fault-tolerant 2-partition in a faulty n-cube.

Theorem 1: For $n \ge 2$, given a set F of n-1 or fewer faulty nodes in an n-cube, any pair of adjacent 2-cubes X and Y in a fault-tolerant 2-partition based on P is different in some dimension. If the neighboring node y_1(in Y) of the non-faulty node x_1(in X) is faulty, there exist some channels with both end nodes being non-faulty, and one of the end nodes is the non-faulty *buddy* of node x_1, and the other is in the 2-cube Y. $\square$

From ***Theorem 1***, we know that at most two hops are required to transfer a packet between any two adjacent 2-cubes. If node x_1 cannot send a packet to a desired adjacent 2-cube directly, one of the non-faulty buddies of x_1 can send this packet to the adjacent 2-cube directly. Thus, the result of fault-tolerant 2-partition in faulty n-cubes produces a useful property, that is, any pair of adjacent 2-cubes can exchange packets directly without passing through other 2-cubes(see Figure 3). In the following subsection 3.2, we will propose a scheme for the partition of channels in faulty n-cubes. This scheme is based on the result of the fault-tolerant 2-partition and channel labeling.

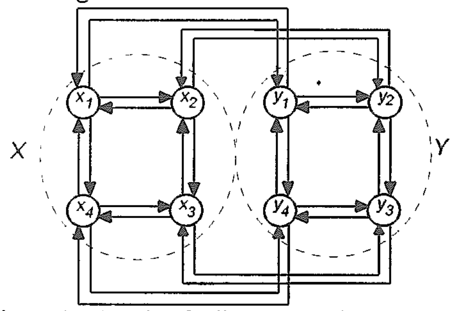

Figure 3. A pair of adjacent 2-cubes X and Y.

3.2 Partitioning of Channels

We say that a faulty n-cube is a ***supercube*** Q'_{n-2} with n-2 dimensions if there exists a fault-tolerant 2-partition based on P for the faulty n-cube. Without loss of generality, we assume that $P=\{0, 1\}$ for the 2-partition in the following. Each ***supernode*** $X'=(x'_{n-1}, x'_{n-2},..., x'_2, *, *)$ in a Q'_{n-2} contains four nodes $(x'_{n-1}, x'_{n-2},..., x'_2, 0, 0)$, $(x'_{n-1}, x'_{n-2},..., x'_2, 0, 1)$, $(x'_{n-1}, x'_{n-2},..., x'_2, 1, 0)$ and $(x'_{n-1}, x'_{n-2},..., x'_2, 1, 1)$, where $x'_i \in \{0, 1\}$ for $2 \le i \le n-1$. Let ***s_n(x)*** denote the supernode that contains node x in a Q'_{n-2}. Each supernode X' in a Q'_{n-2} is assigned a label ***l***$(x'_{n-1}, x'_{n-2},..., x'_2, *, *)$. The assignment of the label to a supernode is based on the order of that supernode in a ***Hamilton path***, where the first supernode in the path is assigned the label 0 and the last supernode in the path is assigned the label 2^{n-2}-1. A Hamilton path visits each supernode in the Q'_{n-2} only once. The ***high-channel network*** contains all the channels which are from lower label supernodes to higher label supernodes, the ***low-channel network*** contains all the channels from higher label supernodes to lower label supernodes, and the ***inter-channel network*** contains all the channels within the supernodes. An example is shown in Figure 4.

Obviously, for any supercube Q'_{n-2}, there exists many Hamilton paths. We can use certain Hamilton path to define the label assignment function $l()$. However, the performance of a routing scheme is dependent on the selection of a Hamilton path[7]. For a supercube Q'_{n-2} in this paper, the label assignment function $l()$ we used for a supernode with address $(x'_{n-1}, x'_{n-2},..., x'_2, *, *)$ is

$$l(x'_{n-1}, x'_{n-2},..., x'_2, *, *)=\sum_{i=2}^{n-1}(c_i\overline{x'_i}2^{i-2}+\bar{c}_i x'_i 2^{i-2}),$$

where $c_{n-1}=0$, $c_{n-j}=d_{n-3}\oplus d_{n-4}\oplus ...\oplus d_{n-j+1}$ for $1<j\le n-2$. Besides, for any node k(not supernode) in Q'_{n-2}, let the $l()$ function for node k be

$$l(k)=l(s\_n(k)),$$

that is node k and its supernode has the same label value. Clearly, the above label assignment function is derived from the encoding method of the ***Gray Code***. Each supernode has its unique label by using such label assignment function. An example for this assignment function in a faulty 4-cube is shown in Figure 5. We assume that nodes 0001, 0111 and 1101 are faulty, and the corresponding fault-tolerant 2-partition has $P=\{0, 1\}$.

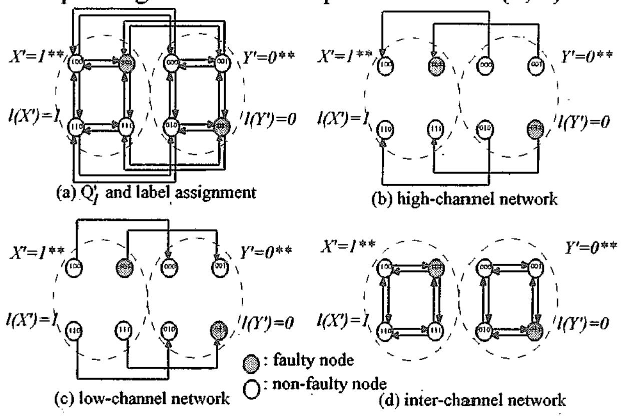

Figure 4. The fault-tolerant 2-partition with $P=\{0, 1\}$ and the corresponding label assignment for a faulty 3-cube with two faulty nodes.

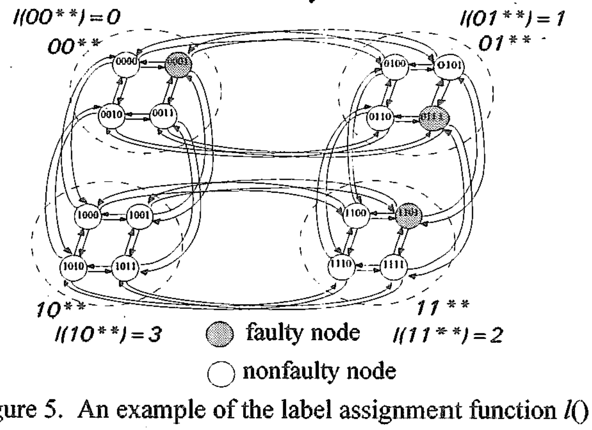

Figure 5. An example of the label assignment function $l()$ in a faulty 4-cube corresponding to a fault-tolerant 2-partition with $P=\{0, 1\}$.

4. Deadlock-Free Multicast for Faulty Wormhole-Routed Hypercubes

4.1 Basic Concept of Deadlock-Free Multicast

In this paper, we propose a path-like, dual-path multicast algorithm. In wormhole networks, communication channels and message buffers constitute the set of permanent reusable resources. Messages are the entities that compete for these resources. Deadlock refers to the situation that in a set of messages each is blocked indefinitely because each message in the set holds some resources also needed by another message. An example of the deadlock in wormhole networks is shown in Figure 6. In general, a circular wait condition should be inhibited to prevent the deadlock. In next subsection, we will present our fault-tolerant multicast algorithm in detail.

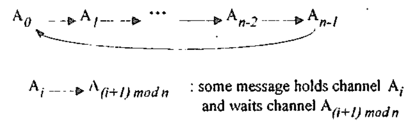
A_0
A_1
A_{n-2}
A_{n-1}
$A_i \dashrightarrow A_{(i+1)\ mod\ n}$: some message holds channel A_i and waits channel $A_{(i+1)\ mod\ n}$

Figure 6. An example deadlock situation in wormhole networks.

4.2 Fault-Tolerant Dual-Path Multicast in Faulty Hypercubes

Let s and D denote the source node and destination set for a multicast in a faulty n-cube, respectively. Note that s and the nodes in D must be non-faulty. The main idea of our dual-path multicast algorithm is to divide the destination set D into two subsets D_1 and D_2 such that D_1 contains all the destination nodes with equal or higher $l()$ value than $l(s)$ and D_2 contains the nodes with lower $l()$ value than $l(s)$. Then the multicast messages from s will be sent to the destination nodes in D_1 through the high-channel and inter-channel networks, and to the destination nodes in D_2 through the low-channel and inter-channel networks.

Obviously, source node s has to do a pre-processing task named ***message preparation*** before starting a mulitcast. The message preparation for node s is to divide the destination node set D into two subsets: D_H and D_L, which are then sorted in ascending order and descending order, respectively, with the label of each supernode that the destination node is within it as the key for sorting. Let D_H be the set $\{dh_1, dh_2, \ldots, dh_r\}$, where $l(s) \le l(dh_i) \le l(dh_j)$, dh_i and $dh_j \in D$ for $1 \le i < j \le r \le 2^n$. Let D_L be the set $\{dl_1, dl_2, \ldots, dl_t\}$, where $l(s) > l(dl_i) \ge l(dl_j)$, dl_i and $dl_j \in D$ for $1 \le i < j \le t \le 2^n$. Then we perform the multicasting for set D_H among some supernodes according to the following traversal order:

$$s\_n(s) \Rightarrow s\_n(dh_1) \Rightarrow s\_n(dh_2) \Rightarrow \ldots \Rightarrow s\_n(dh_r)$$

and the traversal order for set D_L is

$$s\_n(s) \Rightarrow s\_n(dl_1) \Rightarrow s\_n(dl_2) \Rightarrow \ldots \Rightarrow s\_n(dl_t).$$

If we adopt the above approach, there will be a problem: *Does any pair of adjacent supernodes in the traversal path transfer a packet directly without passing through other supernodes?* In ***Theorem 1***, we have proven that any pair of adjacent supernodes can transfer a packet directly with at most two hops. Thus, the above dual-path multicast approach is feasible among supernodes.

The overall transmission process for our approach is illustrated in Figure 7.

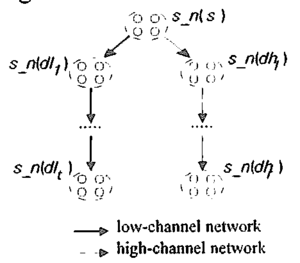

Figure 7. The conceptual illustration of the fault-tolerant dual-path multicast.

By the definition of node and channel partitioning schemes, each of the high-channel and low-channel subnetworks comprises a separate set of channels in the network. Thus, there is no cycle dependency in the high-channel or low-channel network.

However, the deadlock situation will exist within the supernodes if we do not handle the multicast in the inter-channel network properly. The structure graph of a supernode is illustrated in Figure 8(a). Note that the external dimension part of the node label in Figure 8 is ignored because only the internal dimension part is needed to indicate the relationship of nodes in the supernode. Assuming a packet can be delivered through any minimal routing path in the supernode, the corresponding channel dependency graph is shown in Figure 8(b). Since there are two cycles in the channel dependency graph, deadlock is possible. Note that the dependency cycle comprised by two nodes(see Figure 8(c)) is ignored because it is not the minimal routing path. One way to avoid deadlock is to inhibit packets to be forwarded from channel d to a and from e to h. The resulting channel dependency graph is shown in Figure 8(d). It is easy to verify that the routing is still minimal. However, to send a packet from node 00 to node 11, the packet must be forwarded through node 10, as the path through node 01 is no longer permitted. Based on the turn model[11], we still have many ways to select such two inhibited turns.

Thus, the routing algorithm within a fault-free supernode is shown in Figure 9. This algorithm is derived from Figure 8. Pattern ($\$\$ \xrightarrow{p} \#\#$) denotes that node \$\$ sends a packet to node ## through channel p. The ***e-cube routing***(see Figure 10) in the fault-free n-cube was proposed by Sullivan and Bashkow[6] in the design of CHOPP multicomputer. The algorithm uses a function, $f_1(v)$, to indicate the location of the first 1 from right in an n-bit binary number v. If the first 1 of v is absent, let $f_1(v)=n$. For example:

$$f_1(1100)=2 \quad f_1(0001)=0 \quad f_1(0000)=4$$

In addition, e-cube routing can produce a routing path with minimal length. For the routing within a supernode, only the information of the internal dimensions is needed.

So, we modify the *e*-cube algorithm to obtain the algorithm *e*'-2cube(see Figure 11) for Figure 9.

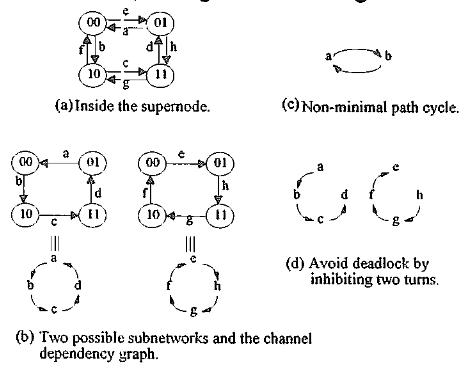
(a) Inside the supernode.
(c) Non-minimal path cycle.
(b) Two possible subnetworks and the channel dependency graph.
(d) Avoid deadlock by inhibiting two turns.

Figure 8. The deadlock problem within a supernode(note that the labels of external dimensions for each node are ignored).

```
Algorithm: Routing within a fault-free supernode
/* Input: local node address v, destination node address u. */
/* Output: The channel needed by node v to transfer the received packet
   if $v \neq u$ */
IF v=u
THEN forward the packet to the local processor; return EMPTY;
ELSE IF the internal dimension parts of v and u are 00 and 11 respectively
      THEN return the channel b(because $00 \xrightarrow{b} 10 \xrightarrow{c} 11$);
      ELSE IF the internal dimension parts of v and u are 11 and 00 respectively
            THEN return the channel g(because $11 \xrightarrow{g} 10 \xrightarrow{f} 00$);
            ELSE call algorithm e'-2cube with parameters v and u;
                  return the output from algorithm e'-2cube;
            END_IF
      END_IF
END_IF
```

Figure 9. The deadlock-free routing within a fault-free supernode.

```
Algorithm: Routing in a fault-free n-cube
/* Input: local node address v, received packet with destination d. */
/* Output: The channel needed by node v to transfer the received packet   if $v \neq d$. */
Step 1: Let $k = f_1(v \oplus d)$.
Step 2: IF k=n
        THEN forward the packet to the local processor;
        ELSE return the channel from node v to the neighboring node $v \oplus 2^k$;
        END_IF
```

Figure 10. *e*-cube routing in a fault-free *n*-cube.

```
Algorithm: e'-2cube routing for Figure 9
/* Input: local node address v, destination node address u. */
/* Output: The channel needed by node v to transfer the received packetif $v \neq u$. */
/* Function $g_i(v,u)$ is the position of the ith internal dimension
from right, that is, the ith internal dimension, at which v and
u differ. If no such dimension exists, $g_i(v,u)=n$. */
Step 1: Let $k = g_1(v,u)$;
Step 2: IF k=n
        THEN forward the packet to the local processor; return EMPTY;
        ELSE return the channel from node v to the neighboring node $v \oplus 2^k$;
        END_IF
/* Note that this channel is the outgoing channel from v for the packet in next transmission. */
```

Figure 11. The *e*'-2cube routing for Figure 9.

From the above proposed node partitioning scheme, we know that each faulty supernode contains exactly one faulty node. The inter-channels connected to the faulty node are useless. The overall view of a faulty supernode is shown in Figure 12. It is easy to see that there exists no channel dependency cycle within a faulty supernode, and the non-faulty nodes constitute a connected graph. Each non-faulty node can communicate with other non-faulty nodes(within the same supernode) through at most two hops. Thus, for the faulty supernode which contains one faulty node, we can develop a deadlock-free routing algorithm(see Figure 13) within it by modifying the *e*-cube routing.

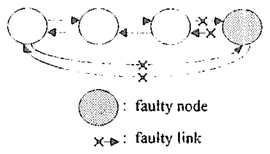

Figure 12. An illustration of a faulty supernode.

```
Algorithm: Routing within a faulty supernode
/* Input: local node address v, destination node address u.*/
/* Output: The channel needed by node v to transfer the received packet if $v \neq u$. */
/* Function $g_i(v,u)$ is the position of the ith internal dimension
   from right, that is, the ith internal dimension, at which v and
   u differ. If no such dimension exists, $g_i(v,u)=n$. */
Step 1:Let $k = g_1(v,u)$;
Step 2: IF k=n
        THEN forward the packet to the local processor; return EMPTY;
        ELSE IF node $v \oplus 2^k$ is non-faulty
              THEN return the channel from node v to the neighboring node $v \oplus 2^k$;
              ELSE return the channel from node v to the neighboring node $v \oplus 2^{g_2(v,d)}$;
              END_IF
      END_IF
```

Figure 13. The deadlock-free routing within a faulty supernode.

After developing the above deadlock-free routing algorithm within a fault-free/faulty supernode, the details of the proposed fault-tolerant deadlock-free dual-path multicast scheme are shown in Figures 16 and 17. We divide the multicast scheme into two parts, one is for the ***source*** node and the other is for the ***intermediate*** node. Figure 16 shows the message preparation and multicast scheme for the source node, and Figure 17 indicates the multicast method in which the multicast decision is made at each intermediate node. In these two algorithms, two functions *R*()(see Figure 14) and *R*'()(see Figure 15) are used to determine the next outgoing channel for the received message.

Theorem 2: The proposed fault-tolerant multicast algorithms in Fig. 16 and Fig. 17 are deadlock-free.

Proof: Because the low-channel network and high-channel network are channel-disjoint, from Fig. 7 we know that the deadlock may only occur within a supernode. Moreover, the deadlock-free approach within a supernode is given in Fig. 8. Therefore, it is clear that the proposed algorithms in Fig. 16 and Fig. 17 are deadlock-free. □

```
Function: R(v, u)
/* v and u are non-faulty nodes. */
IF s_n(v) is a fault-free supernode
  THEN c= the returned channel of the algorithm in Figure 9 with parameters v and u;
  ELSE c= the returned channel of the algorithm in Figure 13 with parameters v and u;
END_IF
return the channel c;
```

Figure 14. The function *R*().

```
Function: R'(v, u)
/* v and u are non-faulty nodes. */
IF l(v)<l(u)
```

```
  THEN $l(p)=\max\{l(w): l(w) \leq l(u)$ and $w$ is a neighboring node of $v\}$;
  ELSE $l(p)=\min\{l(w): l(w) \geq l(u)$ and $w$ is a neighboring node of $v\}$;
END_IF
IF $p$ is a faulty node
  THEN Let $p$ be one of the non-faulty buddies of $v$;
END_IF
return the channel from $v$ to $p$;
```

Figure 15. The function $R'()$.

```
Algorithm: Multicast for the source node
/* Input: $D$ and $s$ denote the destination set and source node, respectively. */
Step 1: Divide $D$ into three sets $D_H$, $D_L$ and $D_E$. Let $D_H$ be the set
        $\{dh_1, dh_2, \dots, dh_r\}$, where $l(s)<l(dh_i) \leq l(dh_j)$, $dh_i$ and
        $dh_j \in D$ for $1 \leq i<j \leq r \leq 2^n$. Let $D_L$ be the set $\{dl_1, dl_2, \dots, dl_t\}$,
        where $l(s)>l(dl_i) \geq l(dl_j)$, $dl_i$ and $dl_j \in D$ for $1 \leq i<j \leq t \leq 2^n$. Let
        $D_E$ be the set $\{de_1, de_2, \dots, de_k\}$, where $l(s)=l(de_i)$, $de_i \in D$ for $1 \leq i \leq k \leq 3$.
/* Note that for any node $v$, $l(v)=l(s\_n(v))$. */
Step 2: IF $D_E \neq \varnothing$ THEN
            $i=1$ and $C=\varnothing$;
            WHILE $i \leq k$ DO /* $C$ is the set of outgoing inter-channels. */
                channel $c_i=R(s, de_i)$;
                $C=C \cup \{c_i\}$;      $i=i+1$;
            END_WHILE
            IF $|C|=1$ THEN /* Only one outgoing inter-channel is needed. */
                Construct message $M_1$ that contains the set $D_E$
                    as part of the header;
                Acquire channel $c_1$;
                Send $M_1$ into the channel $c_1$;
            END_IF
            IF two buddies $b_1$ and $b_2$ of $s$ are both in $D_E$
            /* Two outgoing inter-channels are needed. */
            THEN
            Duplicate two messages, one containing $\{b_1\}$
              as part of the header and the other containing
              $D_E-\{b_1\}$ as part of the header;
            Acquire all channels in $C$;
            Send two messages to the two buddies of $s$
                through channels of $C$, respectively;
            END_IF
        END_IF
Step 3: IF $D_H \neq \varnothing$ THEN /*The set $D_H$ is not empty. */
            $c_0=R'(s, dh_1)$;
            construct the message $M$ that contains $D_H$ in its header;
            Acquire channel $c_0$;
            Send message $M$ into channel $c_0$;
        END_IF
Step 4: IF $D_L \neq \varnothing$ THEN /* The set $D_L$ is not empty. */
            $c_0=R'(s,dl_1)$;
            construct the message $M$ that contains $D_L$ in its header;
            Acquire channel $c_0$;
            Send message $M$ into channel $c_0$;
        END_IF
```

Figure 16. Multicast for the source node.

```
Algorithm: Multicast for the intermediate node
/* Input: A received message with a sorted header of destination list $D_I=\{d_1, \dots, d_k\}$ and
   a local address $w$. */
Step 1: IF $w=d_1$ THEN /* $w$ is the destination node. */
            $D_I'=D_I-\{d_1\}$.
            Forward a copy of the message into the local node;
            IF $D_I'=\varnothing$ THEN exit;
            END_IF
        ELSE $D_I'=D_I$;
        END_IF
Step 2: IF $w=d_1$ THEN   $i=2$;
               ELSE   $i=1$;
        END_IF
        Let $C=\varnothing$ and $D_E=\varnothing$;
        WHILE $i \leq k$ and $l(w)=l(d_i)$ DO /* $C$ is the set of outgoing inter-channels. */
            channel $c_i=R(w, d_i)$;
            $C=C \cup \{c_i\}$;      $D_E=D_E \cup \{d_i\}$;
            $D_I'=D_I'-\{d_i\}$;      $i=i+1$;
        END_WHILE;
        IF $i<k$ THEN    $c_0=R'(w, d_i)$; /* $c_0$ is the outgoing high-channel/low-channel. */
        END_IF
Step 3: IF $C=\varnothing$ and $i<k$
          THEN /* No other destination nodes in the supernode $s\_n(w)$. */
            Acquire channel $c_0$;
            Send the message into channel $c_0$ with destinations $D_I'$ in its header;
            exit;
        END_IF
        IF $|C|=1$ and $i=k$ THEN
        /* $s\_n(w)$ is the last supernode traversed, and one inter-channel is needed. */
                Construct the message containing $D_E$ as part of the header;
                Acquire channel $c_1$;
                Send this message into channel $c_1$;
                exit;
        END_IF
        IF $|C|=1$ and $i<k$ THEN
        /* One outgoing inter-channel and one outgoing high-/low- channel are needed. */
                Duplicate two messages, one containing $D_E$ as part of the
                 header and the other containing $D_I'$ as part of the header;
                 Acquire channels $c_0$ and $c_1$.
                Send two messages into channels $c_0$ and $c_1$, respectively.
                exit;
        END_IF
        IF two buddies $b_1$ and $b_2$ of $w$ are both in $D_E$ and $i=k$ THEN
        /*$s\_n(w)$ is the last supernode traversed, and two outgoing inter-channels are needed.*/
            Duplicate two messages, one containing $\{b_1\}$ as part of the header
              and the other containing $D_E-\{b_1\}$ as part of the header;
            Acquire channels $c_1$ and $c_2$;
            Send two messages into channels $c_1$ and $c_2$, respectively;
            exit;
        END_IF
        IF two buddies $b_1$ and $b_2$ of $w$ are both in $D_E$ and $i<k$ THEN
        /* Two outgoing inter-channels and one outgoing high-/low- channel are needed.*/
            Duplicate three messages, one containing $\{b_1\}$ as part of
            the header, another containing $D_E-\{b_1\}$ as part of the header
            and the other containing $D_I'$ as part of the header;
            Acquire channels $c_1$, $c_2$ and $c_0$,
            Send three messages into channels $c_1$, $c_2$ and $c_0$, respectively;
        END_IF
```

Figure 17. Multicast for the intermediate node.

4.3 An example

An example for the proposed fault-tolerant multicast in a faulty 5-cube is shown in Figure 18. Let the faulty nodes be 00100, 01001, 11110, and 10011. Clearly, $P=\{0, 1\}$ generates one of the fault-tolerant 2-partition for this faulty 5-cube. We assume that the source node and the destination set are 01100 and {00010, 00101, 00111, 01000, 01010, 11000, 11101, 10100, 10001}, respectively. Also, the ordered sets D_L and D_H obtained at the source node 01100 are {00101, 00111, 00010} and {01000, 01010, 11000, 11101, 10100, 10001}, respectively. Thus, the multicast traversal order of D_L among supernodes is

$$011{**} \Rightarrow 001{**} \Rightarrow 000{**},$$

and the traversal order of D_H among supernodes is

$$011{**} \Rightarrow 010{**} \Rightarrow 110{**} \Rightarrow 111{**} \Rightarrow 101{**} \Rightarrow 100{**}$$

The details of the multicast process are depicted in Figure 18.

5. Performance Analysis

We use the criterion "***N*=the number of channels required for a multicast**" to measure the performance of the proposed algorithm. For example, the multicast in Figure 18 can be represented as Figure 19. In such representation, N is equal to the number of channels, that is, 14. Each multicast can be represented as the way of Figure 19, and N is equal to the number of channels.. In the following subsections, we analyze the system performance with both theoretic and simulation ways.

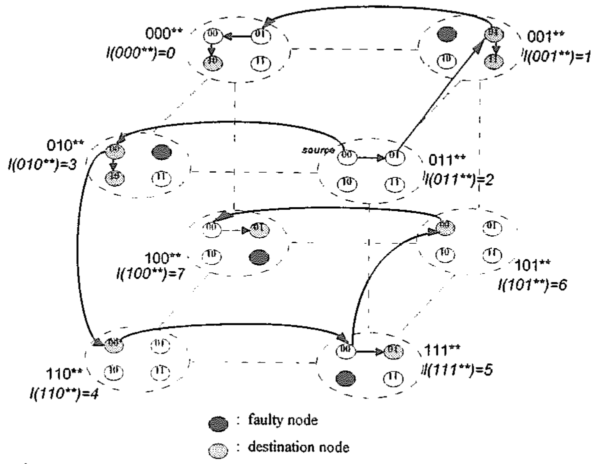

Figure 18. An example for the fault-tolerant dual-path multicasting algorithm in a 5-cube.

5.1 Theoretic Analysis

Based on the above criterion, the best case for the proposed multicast algorithm is that there is only *one* destination node and the destination node is the neighbor of the source node. Under this situation, N is equal to 1. Therefore, the worst case for the proposed algorithm is that the n-cube is fault-free and the destination nodes are other 2^n-1 non-source nodes. Under this situation, N is equal to $2^{n-2}-1$(*the channels needed for muticasting among all supernodes*)+$3\times 2^{n-2}$(*the channels needed for muticasting within all supernodes*)=2^n-1, where $n\geq 2$. However, the average case depends on the label assignment function $l()$, the position of the source node, the number of faulty nodes, the distribution of faulty nodes, the number of destination nodes, and the distribution of destination nodes. Unfortunately, these factors are non-predictable. So, it is difficult to use any theoretic approach to obtain N for the average case. The theoretic results are summarized in Table 1. Table 2 also shows the value $\frac{N}{\# of\ total\ channels}$ for some n's under the theoretic worst-case. We can find that $\lim_{n\to\infty}\frac{N}{\# of\ total\ channels}=\lim_{n\to\infty}\frac{2^n-1}{n\times 2^n}\cong\lim_{n\to\infty}\frac{1}{n}\cong 0$. In other word, the proposed algorithm has good performance when n is large.

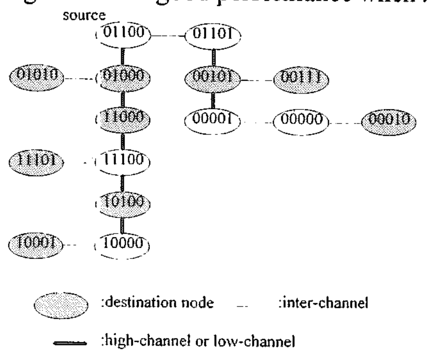

Figure 19: The simplified graph for the multicast in Figure 17.

	N	$\frac{N}{\# of\ total\ channels}$
Best Case	1	$\frac{1}{n\times 2^n}$
Worst Case	2^n-1	$\frac{2^n-1}{n\times 2^n}$

Table 1: The results of the theoretic analysis.

n	$\frac{N}{\# of\ total\ channels}\cong\frac{1}{n}$
4	25%
10	10%
16	6.25%
32	3.13%

Table 2: The results for some n's under the theoretic worst-case.

5.2 Simulation Analysis

From section 5.1, we know that it is difficult to perform the average-case analysis with the theoretic way. In order to analyze the performance for the average case, we use a 10-cube(1024 nodes) to be the experiment model. In this model, the $l()$ function adopts the *Gray Code* assignment, and those factors, the position of the source node, the distribution of faulty nodes, and the distribution of destination nodes are selected randomly. For each multicast, we represent it as that in Figure 19 and use the tree traversal algorithm to determine N. The experimental results are listed in Table 3. For each condition(row), 10 different random distributions of these nodes are taken, then the average value of the 10 results is listed in the column N in Table 3. From the following three sets: the rows with # of destination nodes =64(row 3, row 10), the rows with # of destination nodes =128(row 4, row 7, row 11), and the rows with # of destination nodes =16(row 6, row 9) in Table 3, it shows that the communication overhead caused by the faulty nodes is less than 0.1% in average. Therefore, we can find that the required channel percentage in a 10-cube is at most 2.62% if the number of destination nodes is no greater than 128. From the simulation results for the average case, the proposed fault-tolerant multicast algorithm shows a good performance in a faulty n-cube.

row #	# of faulty nodes	# of destination nodes	N	N / # of total channels
1	0	8	135	1.32%
2	0	32	155	1.51%
3	0	64	190	1.86%
4	0	128	261	2.55%
5	4	4	131	1.28%
6	4	16	145	1.42%
7	4	128	263	2.57%
8	4	256	385	3.76%
9	8	16	146	1.43%

10	8	64	199	1.94%
11	8	128	268	2.62%
12	8	512	642	6.27%

Table 3: The simulation results for the 10-cube model.

5.3 More Than n-1 Faults

In this paper, we only consider the situation that the number of faulty nodes is less than n because there may exist an isolated non-faulty node if the number of faulty nodes is greater than n-1 in an n-cube. Recall that the first step in our algorithm is to find a fault-tolerant 2-partition for the faulty n-cube. From the definition of fault-tolerant 2-partition, it is easy to see that a faulty n-cube with at most 2^{n-2} faults may have a fault-tolerant 2-partition. For example, let the faulty nodes in Figure 18 be 00011, 00100, 01110, 01001, 11010, 11110, 10101, 10011. Thus, the partition with P={0, 1} is still a fault-tolerant 2-partition under such fault distribution and the multicast can still be completed by using the proposed algorithm. Clearly, when the number of faulty nodes is greater than n-1, whether there exists a fault-tolerant 2-partition depends on the distribution of the faulty nodes. Finally, for a faulty n-cube with at most 2^{n-2} faults, the proposed algorithm also works correctly if a fault-tolerant 2-partition for this n-cube can be found.

6. Conclusions

In this paper, a fault-tolerant multicast algorithm for a wormhole routed n-dimensional hypercube with at most n-1 faults is proposed. The exclusive use of channels make deadlock-free the essential element of designing algorithms in wormhole routed networks. This brings about the more complex fault-tolerant multicast design under node/link faults. Previous researches have focused on fault-tolerant one-to-one routing algorithms for n-dimensional meshes. However, little research has been done on fault-tolerant multicast algorithms. The property of subcube partitioning provides a simple and systematic way to develop fault-tolerant multicast algorithms. Our approach is not based on adding physical or virtual channels to the network topology. Instead, we integrate several techniques such as partitioning of nodes, partitioning of channels, node label assignments, and path-like multicast, and have achieved the goal efficiently. Moreover, the simulation results show that the communication overhead caused by the proposed algorithm under faults is very small. Finally, we address that for a faulty n-cube with at most 2^{n-2} faults, the proposed fault-tolerant algorithm also works correctly if a fault-tolerant 2-partition for this n-cube can be found.

Reference

[1] W. J. Dally, "Virtual channel flow control," *IEEE Trans. Comput.*, vol. 3, pp. 194-205, Mar. 1992.

[2] W. J. Dally and C. L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Trans. Comput.*, vol. C36. pp. 547-553, May 1987.

[3] L. M. Ni and P. K. McKinley, "A survey of wormhole routing techniques in direct networks," *IEEE Comput.*, vol. 26, pp. 62-76, Feb. 1993.

[4] X. Lin, P. K. McKinley, and L. M. Ni, "Deadlock-free multicast wormhole routing in 2-D mesh multicomputers," *IEEE Trans. Parallel and Distributed Systems*, vol. 5, no. 8, pp. 793-804, Aug. 1994.

[5] X. Lin and L. M. Ni, "Multicast communication in multicomputers networks," *in Proc. of the 1990 Int'l. Conf. on parallel processing*, vol. III, pp. 114-118, 1990.

[6] H. Sulliran and T. R. Bashkow, "A large scale, homogeneous, fully distributed parallel machine," *in Proc. of the 4th Int'l. Symp. on Computer Architecture*, vol. 5, pp. 105-124, Mar 1977.

[7] X. Lin and L. M. Ni, "Deadlock-free multicast wormhole routing in multicomputer networks," *in Proc. of the 18th Int'l. Symp. on Computer Architecture*, pp. 116-125, 1991.

[8] C. J. Glass and L. M. Ni, "Maximally fully adaptive routing in 2-D meshes," *in Proc. of the 1992 Int'l. Conf. on Parallel Processing*, vol. I, pp. 101-104, Aug. 1992.

[9] X. Lin, P. K. McKinley, and L. M. Ni, "Performance evaluation of multicast wormhole routing in 2-D mesh multicomputers," *in Proc. of the 1991 Int'l. Conf. on Parallel Processing*, vol. 1, pp. 435-442, 1991.

[10] D. H. Linder and J. C. Harden, "An adaptive and fault tolerant wormhole routing strategy for k-ary n-cubes," *IEEE Trans. Comput.*, vol. 40, no. 1, pp. 2-12, Jan. 1991.

[11] J. Bruck, R. Cypher, and D. Soroker, "Tolerating faults in hypercubes using subcube partitioning," *IEEE Tran. Comput.*, vol. 41, no. 5, pp. 599-605, May 1992.

[12] S. C. Wang and S. Y. Kuo, "Fault tolerance in hyperbus and hypercube multiprocessors using partitioning scheme," *in Proc. of the 1994 Int'l. Conf. on Parallel and Distributed System*, pp. 340-347, Dec. 1994.