Interleaved All-to-All Reliable Broadcast on Meshes and Hypercubes

Sunggu Lee, *Member, IEEE*, and Kang G. Shin, *Fellow, IEEE*

***Abstract*— All-to-all (ATA) reliable broadcast is the problem of reliably distributing information from every node to every other node in point-to-point interconnection networks. A good solution to this problem is essential for clock synchronization, distributed agreement, etc. We propose a novel solution in which the reliable broadcasts from individual nodes are interleaved in such a manner that no two packets contend for the same link at any given time—this type of method is particularly suited for systems which use *virtual cut-through* or *wormhole routing* for fast communication between nodes. Our solution, called the *IHC Algorithm,* can be used on a large class of regular interconnection networks including regular meshes and hypercubes. By adjusting a parameter η referred to as the *interleaving distance,* we can flexibly decrease the link utilization of the IHC algorithm (for normal traffic) at the expense of an increase in the time required for ATA reliable broadcast. We compare the IHC algorithm to several other possible virtual cut-through solutions and a store-and-forward solution. The IHC algorithm with the minimum value of η is shown to be *optimal* in minimizing the execution time of ATA reliable broadcast when used in a dedicated mode (with no other network traffic).**

***Index Terms*—Broadcast, fault-tolerance, hypercube, mesh, reliable communication, virtual cut-through, wormhole routing**

I. Introduction

REGULAR mesh structures [5] and binary hypercubes [3], [23] have drawn considerable attention in recent years as an interconnection topology for the processors of a distributed computing system. The fault-tolerance of these types of structures is an important issue as they are increasingly used for critical applications.

In this paper, we address the *all-to-all (ATA) reliable broadcast* problem, in which every node must reliably broadcast its message to every other node. This is essential for implementing several key fault-tolerant algorithms for distributed agreement [9, 18], clock synchronization [17, 19, 21], and distributed diagnosis of intermittently faulty processors [25]. In these algorithms, each non-faulty node must be able to correctly deliver its message to all of the other non-faulty nodes in the system. Let us assume an interconnection network G with connectivity γ and N nodes, of which t nodes are faulty. Given that faulty nodes can behave in any manner whatsoever, Dolev [9] has shown that correct message delivery can be achieved in G if and only if $t \leq \min\{\lceil \frac{\gamma}{2} \rceil - 1, \lceil \frac{N}{3} \rceil - 1\}$. In [22], Rivest *et al.* describe method of appending each message with an authenticated signature. If a signed message is sent from a node u to another node v, then any disruption of the contents of the message will be detected upon receipt by node v. With signed messages, the bound on the number of faulty nodes can be increased to $t \leq \gamma - 1$.

By Menger's Theorem [4], a γ-connected graph has γ node-disjoint paths between any two nodes u and v. It can be shown that to tolerate the maximum number of faulty nodes, every non-faulty node must send its message to every other node through γ node-disjoint paths. (Fewer than γ node-disjoint paths cannot be used because that would imply that a graph with lower connectivity than γ has the same fault tolerance as a γ-connected graph.) All algorithms described in this paper are of this type. To disrupt communication between nodes u and v with $\geq \gamma$ faulty nodes, there must be at least one faulty node in every disjoint path from u to v. Thus, using this type of method, the probability of correct operation is high even when $\geq \gamma$ faulty nodes are present.

In this paper, we present a novel ATA reliable broadcast algorithm, referred to as the *IHC algorithm*, in which the reliable broadcasts from individual nodes are interleaved in such a manner that no two packets ever contend for the same link at any given time; this results in the highly efficient communication algorithm for networks which use *virtual cut-through* [16] or *wormhole routing* [6]—over 68.7 billion packets can be sent and received in less than 2 milliseconds on a hypercube. Following preliminary background discussions in Section II, Section III describes the general class of interconnection networks for which our proposed solution can be used. Section IV describes the proposed solution and its implementation using virtual cut-through. Section V describes possible alternative solutions based on new and existing reliable broadcast algorithms that use virtual cut-through. Section VI analyzes and compares the proposed solution and alternative solutions. We conclude with Section VII.

II. Background

Multicomputer systems that communicate by message passing have traditionally used a store-and-forward routing method [16]. In this method, when a message is sent from a node u to another node v through an intermediate node w, the

Manucscript received April 21, 1992; revised May 19, 1993. This work was supported in part by the Research Institute of Industrial Science and Technology (Korea), by the NSF under Grants MIP-9012649 and MIP-9203895, by NASA unnder Grant NAG-1-1220, and by the ONR under Grants N00014-91-J-1115 and N00014-94-1-0229.

S. Lee is with the Department of Electrical Engineering, Pohang University, Pohang, Kyungbuk, 790-600, Republic of Korea. Email: slee@vision.postech.ac.kr.

K. G. Shin is with the Real-Time Computing Laboratory, Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109-2122, USA. Email: kgshin@eecs.umich.edu.

IEEE Log Number 9216777.

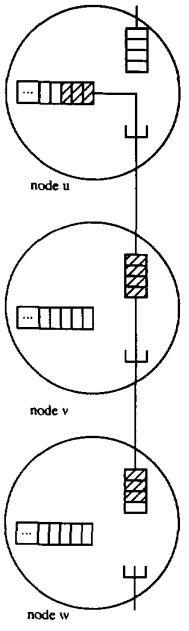


Fig. 1. Illustration of operation of cut-through.

message must be completely stored in w before being passed on. By contrast, in virtual cut-through [16] and wormhole routing [6], instead of storing a message completely in a node and then forwarding it to the next node, the header of the message is advanced directly from incoming to outgoing channels. Only a few control bytes are buffered (in a small on-line FIFO buffer) at each node to determine the out-going channel for the message. Thus, the message becomes spread out across the channels between the source and destination. At an intermediate node, if all outgoing channels are busy, then the entire message is buffered (in a much larger intermediate storage buffer) in virtual cut-through and prevented from moving forward in wormhole routing (i.e., the message is kept in the network). Special routing controller chips have been designed for wormhole routing [6] and virtual cut-through [10]. Deadlock-free wormhole routing is addressed in [7]. The operation of advancing a message immediately from incoming to outgoing channels is referred to as *cut-through*. Virtual cut-through and wormhole routing are collectively referred to as *cut-through (switching) methods.*

The cut-through operation is illustrated by Fig. 1. In this figure, a message of 10 bits originates from node u. At node v, part of the message has been received in a FIFO buffer while the head of the message has already been sent on an outgoing transmitter. At node w, the leading 3 bits of the message have been received in a FIFO buffer. Before the 5th bit arrives at node w, either an outgoing transmitter must be reserved for the message or node w must be prepared to store/receive the message. Thus, of the 10 bits in the message, the first 3 bits are in node w, the middle 4 bits are in node v, and the last 3 bits are in source node u. It is noted that when a message is being cut-through a node such as node v in Fig. 1, it is possible for node v to also *receive* the message by copying the message as it passes through the FIFO buffer. This capability is present in the HARTS routing controller chip [10] and involves a "tee" operation in which the bits of the message are latched into the receiving node as they pass through the FIFO buffer.

Most previous work on reliable broadcast [20] and ATA reliable broadcast [12] have implicitly assumed a store-and-forward routing method. These algorithms can be described by considering the broadcast to be done in several steps and specifying the point-to-point communication patterns that occur at each step. Then the objective is to minimize the total time required for the broadcast operation by using a minimal number of communication steps. Ramanathan and Shin's reliable broadcast (RS) algorithm [20] requires $\gamma + 1$ steps on a hypercube of dimension γ if each node can simultaneously use all of its outgoing links. Fraigniaud's ATA reliable broadcast (FRS) algorithm for hypercubes [12] simply involves executing the RS [20] algorithm at each node in lock step. In every step after the first, each node must merge two messages from the previous step before sending the larger message in the current step. In the last step, the message formed after merging can be made a little bit shorter by removing the portion of the message that would be returned to the originator of that portion of the message. The time required for the FRS [12] algorithm is $(\gamma+1)\tau_S + (2^\gamma - 1)L\tau_L$, where τ_S is the message startup time, L is the message length in bits, and τ_L is the propagation time per bit. To achieve this time, 100% of the link capacity must be used for the entire duration of the ATA broadcast operation.

Recently, there has been work on broadcast algorithms that take advantage of the faster communication possible using cut-through switching methods [15]. In this case, to minimize the total execution time, we must not only minimize the total number of communication steps, but maximize the number of communication steps that can be implemented with cut-through. Kandlur and Shin's reliable broadcast (KS) algorithm [15] is an efficient algorithm for a regularly wrapped hexagonal mesh topology which uses virtual cut-through. In the KS [15] algorithm, the longest path has 2 $\times$ (diameter of mesh) send-receive operations, among which all but three can use cut-through. The analysis done in [15] shows that for a single reliable broadcast operation, the KS [15] algorithm is much faster than an algorithm based on the use of edge-disjoint Hamiltonian cycles (HC's).

III. Interconnection Networks

This section formally presents the class of interconnection networks for which the IHC algorithm for ATA reliable broadcast can be used.

Due to its potential for high-reliability, a point-to-point interconnection network is commonly used to connect the set of processing nodes of a large distributed system. For the purposes of analyzing routing and broadcast algorithms, it is convenient to represent the system by an undirected graph in which the vertices (or nodes) correspond to the processing nodes and the edges correspond to the communication links in the interconnection network. If directed communication links are used, each edge corresponds to 2 communication links.

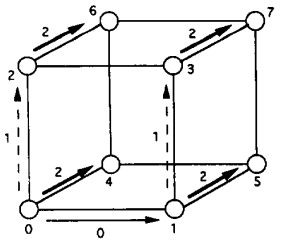


Fig. 2. Q_3, a hypercube of dimension 3.

Thus, given an undirected graph G, the corresponding directed graph G^{dir} is defined to be the graph G with every undirected edge replaced by two directed edges, one in each direction.

A graph is said to be *γ-regular* if all nodes in the graph have degree γ. A *regular* graph is one that is *γ-regular* for some γ [4]. A graph G is said to belong to the class Λ if the following two conditions are satisfied:

LC1: G is *γ-regular* for an even interger γ.
LC2: There are $\frac{\gamma}{2}$ undirected edge-disjoint HC's in G.

The IHC algorithm for ATA reliable broadcast can be used on any graph in the class Λ. (While condition **LC 1** is **not** an essential requirement for the algorithm, it is necessary for the optimality analysis in Section 6, Theorem 4.) Note that if G belongs to the class Λ, they γ is the connectivity of G.

A. Hypercube

An m-dimensional hypercube, denoted by Q_m, has $N = 2^m$ nodes and $m2^{m-1}$ edges. If directed communication links are used, Q_m^{dir} has $m2^m$ directed edges. A Q_m is recursively defined as: (1) Q_0 is a single point, and (2) $Q_m = K_2 \times Q_{m-1}$, where K_2 is a complete graph of 2 nodes and $\times$ denotes the product operation on two graphs [4]. The recursion for Q_m can alternatively be written as $Q_m = Q_{\lceil \frac{m}{2} \rceil} \times Q_{\lfloor \frac{m}{2} \rfloor}$. Each node in a Q_m is uniquely represented by an m-bit address such that the addresses of adjacent nodes differ in exactly one bit. The bits in an address are referred to in right to left order from 0 to $m-1$. Two adjacent nodes which differ in th ith bit will be said to be in *direction* $i\,(0 \le i \le m-1)$ with respect to each other. Fig. 2 shows an example of a Q_3.

Hypercubes of even dimension belong to the class Λ. Condition LC1 is satisfied because a Q_m is m-regular with degree $\gamma = m$. Condition LC2 is satisfied by Theorem 1 below. Let C_k denote an undirected cycle of length k. Then Theorem 1 can be proven with the aid of the following two lemmas.

Lemma 1: [11] $C_k \times C_l$ can be decomposed into 2 undirected HC's $(k, l \ge 3)$.

Lemma 2: [2] The cartesian sum $G + C$, where G is decomposed into 2 undirected HC's and C is a HC, can be decomposed into 3 undirected HC's $(k, l, r \ge 3)$.

Theorem 1: [13] *A* Q_{2k} contains k undirected edge-disjoint HC's.

Proof: The theorem can be proven by induction on $2k$. For the induction basis, a Q_2 is a cycle and Fig. 3 shows 2 undirected edge-disjoint HC's in a Q_4 (although Fig. 3 shows a square mesh, it is also a Q_4 since a Q_4 can be redrawn as a 4×4 torus-wrapped square mesh). Assume a Q_{2l} contains l undirected edge-disjoint HC's for all $l \le k$. We must show that a Q_{2k+2} contains $k+1$ undirected edge-disjoint HC's.

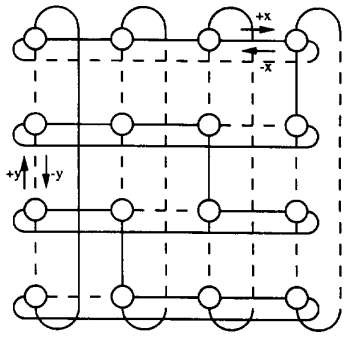


Fig. 3. Torus-wrapped square mesh.

If $k+1$ is even, then decompose the Q_{2k+2} into two Q_{k+1}'s (where decomposition is the inverse of the $\times$ operation). By induction, each Q_{k+1} contains $(k+1)/2$ undirected edge-disjoint HC's. Create $(k+1)/2$ graphs by multiplying (applying the product operation on) the i-th HC's in the two decomposed hypercubes, for all $1 \le i \le (k+1)/2$. By Lemma 1, each product graph contains two undirected edge-disjoint HC's. Then, since all product graphs are edge-disjoint, there exist $(k+1)$ undirected edge-disjoint HC's in the Q_{2k+2}.

If $k+1$ is odd, then decompose the Q_{2k+2} into a Q_k and a Q_{k+2}. Create $k/2-1$ graphs by multiplying the i-th HC's in the two decomposed hypercubes, for all $1 \le i \le k/2-1$. By Lemma 1, each product graph again contains 2 undirected edge-disjoint HC's. There remains one unused HC (C_1) in Q_k and 2 unused HC's $(C_2 + C_3)$ in Q_{k+2}. From Lemma 2, we know that $(C_2 + C_3) \times C_1$ can be decomposed into 3 HC's. Thus, there are a total of $(k+1)$ undirected edge-disjoint HC's in the Q_{2k+2}. □

Theorem 1 can be used to construct the k undirected edge-disjoint HC's in a Q_{2k}. Theorem 1 and Lemmas 1 and 2 use inductive proofs. For Lemmas 1 and 2, Foregger [11] and Aubert and Schneider [2] give examples of the construction method for the basis cases of the inductive proofs and shows how to construct the HC's for larger graphs based upon the solutions for smaller graphs. To construct the k undirected edge-disjoint HC's for a given Q_{2k}, we need to start from the induction basis and use the inductive construction method to build up to the desired hypercube. Although this is clearly a tedious process, it only needs to be done once for a given size hypercube. In addition, results obtained for smaller sized hypercubes can be used in the construction of the HC's for larger sized hypercubes. We do not know of any method to directly construct the k undirected edge-disjoint HC's for a Q_{2k}.

For completeness, the case of hypercubes of odd dimension is considered. It is shown in Theorem 2 that a Q_{2k+1} contains k undirected edge-disjoint HC's. Thus, if one link incident on each node of the Q_{2k+1} is deleted, the resulting graph, with connectivity $\gamma = 2k$, also belongs to the class Λ.

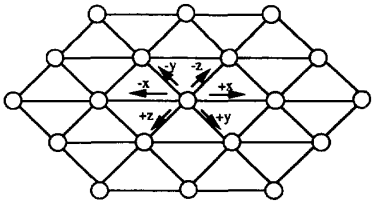


Fig. 4. Hex-mesh of size 3 without wrapping.

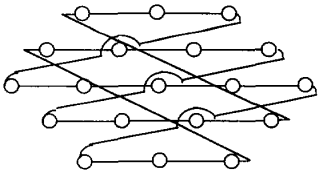


Fig. 5. Hex-mesh with C-type wrapping in $+x$ direction.

Theorem 2: A Q_{2k+1} contains k undirected edge-disjoint HC's.

Proof: A Q_3, with $k = 1$, has the structure of a cube. By using the Gray-code sequence of the node addresses, we can easily obtain 1 undirected edge-disjoint HC. Assume a Q_{2k+1} contains l undirected edge-disjoint HC's for all $l \leq k$. We must show that a Q_{2k+3} contains $k+1$ undirected edge-disjoint HC's.

Suppose $k+1$ is even. Decompose Q_{2k+3} into a Q_{k+1} and a Q_{k+2}. From Theorem 1, Q_{k+1} contains $(k+1)/2$ edge-disjoint HC's. By the induction hypothesis, Q_{k+2} also contains $(k+1)/2$ edge-disjoint HC's. Create $(k+1)/2$ graphs by multiplying the i-th HC's in Q_{k+1} and Q_{k+2}, for all $1 \leq i \leq (k+1)/2$. By Lemma 1, each product graph contains 2 undirected edge-disjoint HC's. Then, since all product graphs are edge-disjoint, Q_{2k+3} contains $k+1$ undirected edge-disjoint HC's.

Suppose $k+1$ is odd. Decompose Q_{2k+3} into a Q_{k+1} and a Q_{k+2}. By induction Q_{k+1} has $k/2$ edge-disjoint HC's. Also, by Theorem 1, Q_{k+2} has $(k+2)/2$ edge-disjoint HC's. Create $k/2-1$ graph by multiplying the i-th HC's in Q_{k+1} and Q_{k+2}, for all $1 \leq i \leq k/2-1$. From lemma 1, each product graph again contains 2 undirected edge-disjoint HC's, for a total of $k-2$ HC's. There remains 1 unused HC in Q_{k+1} and 2 unused HC's in Q_{k+2}. From Lemma 2, we know that the product of these 2 graphs can be decomposed into 3 HC's. □

Alternate proofs of Theorems 1 and 2 are given in [1].

B. Torus-Wrapped Square Mesh

The torus-wrapped square mesh, shown in Fig. 3, belongs to the class Λ. Let SQ_m be a torus-wrapped square mesh of size m, where m is the number of nodes in a single row or column. Since the degree of every node in a SQ_m is 4, condition LC1 is satisfied with $\gamma = 4$. The dashed lines and solid lines in Fig. 3 show two undirected edge-disjoint HC's in a SQ_4, thus satisfying condition LC2. A similar pattern can be used to find two undirected edge-disjoint HC's for any SQ_m.

C. C-Wrapped Hexagonal Mesh

A hexagonal mesh (hex-mesh) has the general structure shown in Fig. 4, which is an unwrapped hex-mesh of size 3. In order to achieve regularity and homogeneity such that identical hardware, software and protocols can be applied uniformly over the network, it is required that the nodes on the hexagonal periphery be wrapped around systematically.

A general systematic method for wrapping hex-meshes is defined in [5] as *C-type wrapping* (see, e.g., Fig. 5). In a C-wrapped hex-mesh, there are six oriented directions, as shown in Fig. 4. Several topological properties of hex-meshes with the C-type wrapping are given in [5]. From these properties, it can easily be derived that the set of edges in any direction of a C-wrapped hex-mesh of size m, denoted as H_m, describes a HC [15]. Thus, there are three undirected edge-disjoint HC's in a H_m; in addition, since a H_m is also γ-regular with $\gamma = 6$, it belongs to the class Λ.

IV. Proposed Solution

The proposed ATA reliable broadcast solution is described for all interconnection networks in the class Λ. Let G be the undirected graph representing any such interconnection network. G is a γ-regular graph (γ even) with $\frac{\gamma}{2}$ undirected edge disjoint HC's. In G^{dir}, there are γ directed HC's HC_1, $\text{HC}_2, \cdots, \text{HC}_\gamma$. For a given node v, $next_i(v)$, and $prev_i(v)$ denote the nodes immediately following and preceding node v in HC_i. Let us arbitrarily designate a node as N_0. For any node $v, \text{ID}_j(v)$ is the distance from N_0 to v when traversing HC_j. N is the total number of nodes. The notation $[x]_y$ is used to denote x mod y, the remainder after x is divided by y. The interleaving distance η is the spacing between nodes that are initiating packets in one iteration of the outermost loop in the proposed solution, described below as algorithm IHC.

IHC Algorithm:

For $i = 0$ **to** $\eta - 1$ **do**
  begin
    for $j = 1$ **to** γ **doparallel**
      for every node v **doparallel**
        if $([\text{ID}_j(v)]_\eta = i)$ **then**
          v sends its message to $next_j(v)$;
    for $j = 1$ **to** $N - 1$ **do**
      for $k = 1$ **to** γ **doparallel**
        for every node v **doparallel**
          begin
            receive message from $prev_k(v)$; $\{*\}$
            if $(j < N - 1)$ **then**
              relay message to $next_k(v)$; $\{*\}$
          end;
end

The IHC algorithm is performed in η stages. Fig. 6 shows an example of the execution of the algorithm in direction HC_j assuming $\eta = 3$. Broadcast messages are sent in fixed-size packets. In stage $i\,(0 \leq i \leq \eta - 1)$, every η-th node in direction HC_j starting from the i-th neighbor of N_0 in directon HC_j is permitted to initiate a packet along HC_j for every $j\,(1 \leq j \leq \gamma)$. Referring to Fig. 6, in a given HC, the nodes numbered i initiate packets in stage i. η can be considered as

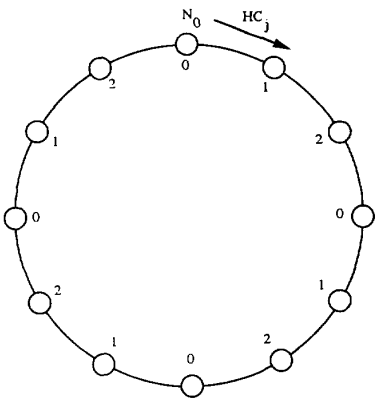


Fig. 6. Nodes initiating packets in one HC of the IHC algorithm.

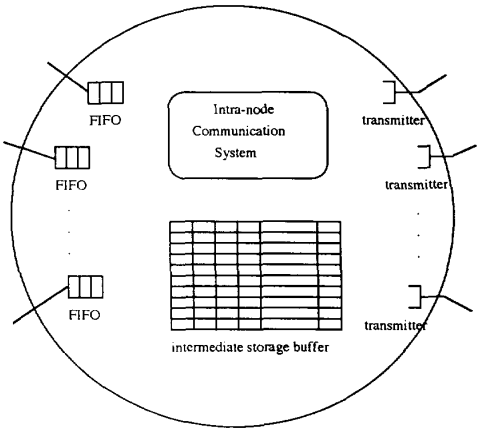


Fig. 7. Node architecture for virtual cut-through.

the interleaving distance. Once packets have been started along directed HC's, they keep flowing for $N-1$ hops along the cycles in which they started. If there are no packets generated by other tasks in the network, then the two steps indicated by "{*}" correspond to a single cut-through operation at each node.

The IHC algorithm has been described assuming that each node can use all of its incoming and outgoing links concurrently. With this assumption, the degree of the network γ does not effect the execution time of the IHC algorithm. However, since γ copies of every message are delivered to every node through edge-disjoint paths, the reliability (and degree of fault tolerance) of the algorithm increases with increasing γ. By using more stages, the algorithm can easily be modified for systems in which each node can use only a subset of its incoming and outgoing links concurrently. For instance, if each node can use only one incoming link and one outgoing link concurrently, then γ sequential invocations of the IHC algorithm can be used (one for each directed HC HC_j). Note that, in this case, it is a simple matter to reduce the execution time (and reliability) of the ATA reliable broadcast by using $k < \gamma$ sequential invocations of the IHC algorithm for k of the γ directed HC's.

We now address the implementation of the IHC algorithm using virtual cut-through. The only difference in the wormhole routing implementation is that blocked packets are not buffered but kept in the network. Note that deadlock does not occur if Dally and Seitz's method of *virtual channels* [7] is used for deadlock prevention. To prevent extremely long "lines of packets" from being formed, however, the wormhole routing implementation of the IHC algorithm must be used in a dedicated mode in which the entire network (or one channel on each directed link) is dedicated to the ATA reliable broadcast operation for the duration of the broadcast operation.

Let us use an architecture similar to the routing controller chip for HARTS [10], a 19-node (H_3) version of which is currently being built at the Real-Time Computing Laboratory. A crucial feature in the HARTS routing controller chip is that all incoming and outgoing links (receivers and transmitters) *can* be used simultaneously. We also assume such a capability in our architecture, shown in Fig. 7.

In high bandwidth communication networks, it has been observed that a large portion (about 80%) of the communication latency is spent in the processing at the transmitters and receivers [14]. (The delay caused by the actual transmission accounts for only 20% of the latency.) Virtual cut-through can be seen as an attempt to eliminate much of the processing (i.e., store-and-forward) at the intermediate nodes between a source and a destination. Thus, when an incoming message cuts through a node, only a very small amount of processing (with the aid of special hardware) is required to determine where and how to advance the message. In Fig. 7, the length of the FIFO buffer is determined by the minimum number of bytes that must be seen and the additional delay required to determine the outgoing transmitter. Many applications in which ATA reliable broadcast is required, such as clock synchronization and distributed diagnosis of intermittently faulty processors, have very short messages that must be broadcast (such as a single clock value or γ bits). For other applications, the messages can be split into fixed size packets. Thus, for the IHC algorithm, we use a packet size of $\mu \times B_{\text{FIFO}}$, where B_{FIFO} is the size of the FIFO buffers at the receivers and μ is a small integer greater than or equal to 1.

Under ideal conditions, in which all nodes can operate perfectly synchronously and there are no other messages in the network, the IHC algorithm can be executed with $\eta = \mu$ (assuming N *modulo* $\mu = 0$). In this case, all possible cut-throughs in the IHC algorighm can occur. Note that $\eta < \mu$ cannot be used because in this case, once a message has been entered into the network from every η-th node, the network (consisting of the FIFO buffers in the receivers of the nodes) cannot "hold" all of the messages. As a result, most of the messages will have to be temporarily stored at the intermediate nodes, defeating the purpose of using cut-through switching.

It network conditions are less than ideal, then we must use an interleaving distance η that is greater than μ. With $\eta > \mu$, we do not require strict synchronization and normal network traffic can be accommodated. The choice of η depends on the degree of link utilization (by the ATA reliable broadcast operation) desired and the degree of synchronization available. Let us refer to the transmission of packets in cycle HC_j during

stage i of the IHC algorithm as an HC_j^i-cycle. Note that even if nodes start sending packets in a cycle HC_j "out of sequence" (possibly due to synchronization inaccuracies), it merely affects the amount of time required and does not affect the correct execution of the algorithm. Also, if normal network traffic or synchronization inaccuracies cause one HC_j^i-cycle to complete before the other HC_k^i-cycles ($k \neq j$), then the nodes on cycle HC_j can start on stage $i+1$ immediately.

There are several ways in which nodes can determine when to stop relaying packets flowing in any given cycle HC_j. One method is to count the number of packets which have been passed. Another method is to add the address of the last node w (with respect to node v) to node v's packet (in the space normally reserved for the routing tag). Then, instead of counting packets, node w simply checks the address of each HC_j broadcast packet to determine if it should stop relaying the packet.

TABLE I
COMMUNICATION PATTERNS FOR EXAMPLE 1

Col. # / Step #	1	2	3	4	5	6	7	8
1	0→1 0→2 0→4 0→8							
2	1→3 2→6 4→12 8→9							
3	3→7 6→14 12→13 9→11	1→5 2→10 4→5 8→10						
4	7→15 14→15 13→15 11→15	5→13 10→11 5→7 10→14	3→11 6→7 12→14 9→13	1→9 2→3 4→6 8→12				
5	15→14 15→13 15→11 15→7	13→12 11→9 7→3 14→6	11→10 7→5 14→10 13→5	9→8 3→1 6→2 12→4	7→6 14→12 13→9 11→3	5→4 10→8 5→1 10→2	3→2 6→4 12→8 9→1	**1→0** **2→0** **4→0** **8→0**

V. RELIABLE BROADCAST WITH VIRTUAL CUT-THROUGH

An ATA reliable broadcast algorithm can be described by first presenting the reliable broadcast algorithm for a single node and then showing how all nodes execute this reliable broadcast algorithm. Two methods have previously been proposed for converting a reliable broadcast algorithm into an ATA reliable broadcast algorithm. In the first method, every node executes the reliable broadcast algorithm concurrently and in lock step [12]. In general, this uses 100% of the available link capacity and may require merge and possibly even split operations. While such operations can easily be done with store-end-forward switching, they would be much more difficult to implement with cut-through switching.

In the second method, each node executes the reliable broadcast algorithm in turn, with the reliable broadcast for one node starting when the previous node finishes. This method leaves a certain amount of unused link capacity (for use by other tasks) and can be executed with cut-through switching methods if the individual reliable broadcast operations can use cut-through. This method can be considered as an alternative to our proposed solution. Thus, in this section, we describe the VRS algorithm, which is the RS [20] algorithm modified to use cut-through. We also describe the KS [15] algorithm and the VSQ algorithm, a cut-through reliable broadcast algorithm for a SQ_m. These three algorithms can form the basis for ATA reliable broadcast in hypercubes, hex-meshes, and square meshes. VRS-ATA (KS-ATA, VSQ-ATA) is define to be the ATA reliable broadcast algorithm in which the VRS (KS [15], VSQ) algorithm is executed for each node in turn.

A. Hypercube Algorithm

The RS [20] algorithm is based on the *recursive doubling* algorithm for broadcast in a hypercube, which is a common broadcast algorithm for hypercubes that can be found in several places in the literature including [20, 24]. In this broadcast algorithm, when a node such as node 0 in Fig. 2 wishes to broadcast a message, it first sends the message in direction 0 to node 1. Nodes 0 and 1 then send the message in direction 1 to nodes 2 and 3, respectively. In the i-th step, all nodes that received the message in step $i-1$ send the message in direction $i-1$. The broadcast completes in n steps in a Q_n.

In the RS [20] algorithm, the node that wishes to broadcast a packet sends a copy of the packet to all of its neighbors. Each of its neighbors then simultaneously execute the recursive doubling algorithm. This algorithm requires $\gamma+1$ (2γ) steps if each node can use all (only one) of its outgoing communication links at a time. It was proven that if the RS [20] algorithm is used, each node receives γ copies of the packet through γ disjoint paths in the fault-free situation [20]. This statement can also be proven by observing that the RS [20] algorithm uses γ edge-disjoint spanning trees as in [12]. The execution of the RS [20] algorithm is illustrated with Example 1.

Example 1: Suppose node 0 wishes to reliably broadcast a packet to all other nodes in a Q_4. The send-receive operations that occur at each step of the algorithm are shown in Table I. The send-receive operations shown in bold in Step 5 of the algorithm (column 8) can be optionally omitted since they simply return copies of the packet to their originator.

To convert the RS [20] algorithm into an efficient cut-through algorithm, we need to convert as many store-and-forward operations as possible into cut-through operations. When a node u receives a packet from direction i and then *immediately* sends it on in direction $[i+1]_m$, u will be said to have *forwarded* the packet. When a node u sends a packet in direction i and *later* sends a copy of the same packet in direction $[i+1]_m$, u will be said to have *redirected* the packet. Clearly, every time a node forwards a packet, a cut-through operation can be used at that node. Also, every time a node has to initiate or redirect a packet, it cannot be done with cut-through. In Table I, the send-receive operations have been separated into columns. In a given column, all of the send-receive operations except the first and last ones are operations in which the packet is being forwarded to the next node. Whenever a new column is started, a redirection is taking place.

In the VRS algorithm (RS [20] algorithm modified to use cut-through), all send-receive operations in which a packet is forwarded is implemented as a cut-through operation. All send-receive operations in which a packet has to be redirected

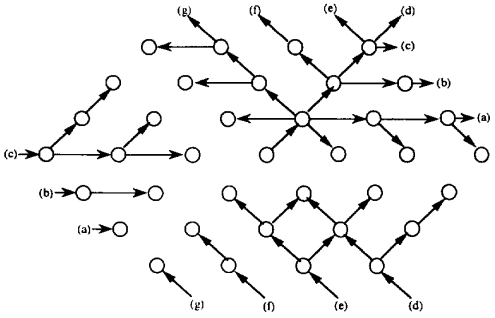


Fig. 8. KS [15] algorithm initiated in one direction.

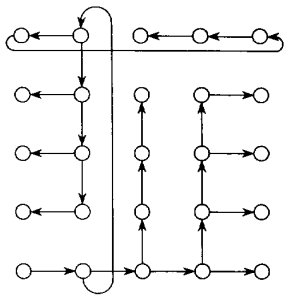


Fig. 9. VSQ algorithm initiated in one direction.

is implemented as a store-and-forward operation. The longest path in the VRS algorithm consists of $\gamma - 1$ store-and-forward operations and 2 cut-through operations (since there is no redirection of the packets received in Step 2 of Table I).

B. C-Wrapped Hex-Mesh Algorithm

The KS [15] algorithm is an efficient cut-through reliable broadcast algorithm for a H_m. When a node v wishes to perform a reliable broadcast, it initiates a copy of the broadcast packet in each of the six possible directions. The pattern of cut-through and store-and-forward operations are identical for each of the six directions. This pattern is shown for one direction in Fig. 8. Every time there is a fork in the pattern, at most one of the paths can use cut-through and the rest of the paths must initiate a new copy of the message, i.e., use a store-and-forward operation. In the KS [15] algorithm, cut-through is used on the path in which the message is propagated on the same direction it arrives on. Then, the longest path with the number of places using store-and-forward requires the longest time to complete. By inspection of Fig. 8, it can be seen that the longest path consists of 3 store-and-forward operations and $2m - 5$ cut-through operations. Since $3m(m-1)+1 = N, \sqrt{\frac{N-1}{3}} < m < \sqrt{\frac{N-1}{3}} + 1$. Thus, a lower bound estimate of the number of cut-through operations required by the KS [15] algorithm is $2\sqrt{(N-1)/3} - 5$.

C. Torus-Wrapped Square Mesh Algorithm

The VSQ algorithm is a cut-through reliable broadcast algorithm for a SQ_m, an $m \times m$ torus-wrapped square mesh. This algorithm is similar to the KS [15] algorithm. When a node v wishes to perform a reliable broadcast, it initiates a copy of the broadcast packet in each of the four possible directions. Then the pattern of cut-through and store-and-forward operations are identical for each of the four directions. When designing this pattern for one direction, we should insure that the patterns for any of the other directions do not "interfere" with the patterns for that direction. Interference would correspond to two arrows pointing in the same direction over the same link, which would mean that one packet could block the other's progress. A pattern satisfying this condition for one direction is shown in Fig. 9. The interpretations of this pattern is the same as for the KS [15] algorithm. The longest path length consists of 3 store-and-forward operations and $2\sqrt{N} - 6$ cut-through operations.

VI. Comparative Analysis

In this section, the IHC algorithm is compared to other possible ATA reliable broadcast algorithms. In the analysis, we will distinguish between the utilization of the available communication links by the ATA reliable broadcast operation and by all other tasks. The latter is referred to by ρ. $\rho = (\rho = 1)$ represents an unloaded (completely congested) system. To achieve $\rho = 0$, we must dedicate the entire network (or one channel on each directed link) to the ATA reliable broadcast operation. This is referred to as a *dedicated network*. Using times quoted for the TORUS wormhole routing chip [6], it is argued in Section VI-A that the IHC algorithm *can* be used in such a dedicated mode even in the largest currently available multicomputer systems. Even in the case of normal network load, our analysis shows that the IHC algorithm compares favorably to the alternate ATA reliable broadcast algorithms.

In sending a packet from a node v to its neighbor w, there are several time parameters of interest. Let L be the length of the packet in bytes and τ_L be the message transmission time in bytes/second. If the packet cuts through node v, then the delay experienced is denoted by α, which is proportional to B_{FIFO}. If the packet is stored into the intermediate storage buffer before being transmitted, then the delay experienced is $\tau_S + L\tau_L$. Since a packet has fixed length and fits into exactly μ FIFO buffers, $L\tau_L = \mu\alpha$. If the packet has to wait because the transmitter to node w is busy, then it will experience the additional queueing delay, D.

A. Dedicated Network

Let us consider the ATA reliable broadcast operation with $\rho = 0$. The times required by the IHC, VRS-ATA, KS-ATA, SQ-ATA, and FRS [12] algorithms are shown in Table II. The execution time for the FRS [12] algorithm is the same as in [12] with a change in notation. For the IHC algorithm, there are η states, each of which requires one message startup and $N-2$ cut-through operations. For the VRS-ATA, KS-ATA, and VSQ-ATA algorithms, there are N reliable broadcast operations, each of which takes the amount of time shown in parentheses.

The IHC algorithm performs better than all of the other cut-through algorithms if $\eta \leq \min\{\log_2 N - 1, 2\sqrt{\frac{N-1}{3}} -$

TABLE II
EXECUTION TIMES WITH $\rho = 0$

Algorithm	Execution Time
IHC	$\eta(\tau_S + \mu\alpha + (N-2)\alpha)$
VRS-ATA	$N((\log_2 N - 1)(\tau_S + \mu\alpha) + 2\alpha)$
KS-ATA	$N(3(\tau_S + \mu\alpha) + (2\sqrt{(N-1)/3} - 5)\alpha)$
VSQ-ATA	$N(3(\tau_S + \mu\alpha) + (2\sqrt{N} - 6)\alpha)$
FRS [10]	$(\log_2 N + 1)\tau_S + (N-1)\mu\alpha$

TABLE III
EXECUTION TIMES WITH $\rho = 0$ AND $\eta = \mu = 2$

Algorithm	Execution Time
IHC	$2\tau_S + 2N\alpha$
VRS-ATA	$N((\log_2 N - 1)(\tau_S + 2\alpha) + 2\alpha)$
KS-ATA	$N(3(\tau_S + 2\alpha) + (2\sqrt{(N-1)/3} - 5)\alpha)$
VSQ-ATA	$N(3(\tau_S + 2\alpha) + (2\sqrt{N} - 6)\alpha)$
FRS [12]	$(\log_2 N + 1)\tau_S + 2(N-1)\alpha$

$2, 2\sqrt{N} - 3\}$. Since $\eta \geq \mu$ and μ can be assumed to be a small fixed constant, this condition is easily achieved. If, in addition, $\eta = \mu$ and $\tau_s \geq \frac{1}{2}\mu^2\alpha$, the IHC algorithm is also faster than the FRS [12] algorithm. Theorem 4 further states that the IHC algorithm with $\eta = \mu = 1$ has the *optimal* computation time.

Theorem 4: Given $\rho = 0$, Algorithm IHC with $\eta = \mu = 1$ is optimal in execution time.

Proof: Since there are N nodes, each of which must receive γ copies of packets from all other nodes, a total of $\gamma N(N-1)$ packets must be sent and received. If the work of sending these packets is divided evenly among the N nodes, each node is responsible for sending $\gamma(N-1)$ packets. In a γ-regular graph, each node has γ outgoing links. For a given node u, if the work of sending the $\gamma(N-1)$ packets is divided evenly among the γ outgoing links, a minimum execution time of $\tau_S + (N-1)\alpha$ is required. Substituting $\eta = \mu = 1$ into the equation for the IHC algorithm in Table III, this is the execution time of the IHC algorithm. □

Although the above theorem states that the IHC algorithm is optimal if $\eta = \mu = 1$, the IHC algorithm can be modified to execute slightly faster for the general case of $\eta = \mu$. Let us assume $\eta = \mu$ and consider only one HC in the description of the IHC algorithm in Section IV. There are $\eta = \mu$ stages, and each stage requires a time of $\tau_S + \mu\alpha + (N-2)\alpha$, resulting in the total execution time of $\eta(\tau_S + \mu\alpha + (N-2)\alpha)$. However, the second stage can start $\mu - 1$ time steps before the first stage completes. The reason for this is that when a node initiates a packet, part of the packet is in the source node queue (as in Fig. 1) for the first $\mu - 1$ time steps. Likewise, when a single stage of the IHC algorithm completes, part of each packet is in the receiving buffer of a node in the last $\mu - 1$ time steps. This method of overlapping stages can be used in every stage after the first (a total of $\mu - 1$ stages). Thus, the execution time of the algorithm is reduced to $T_{\text{IHC}} - (\mu - 1)^2\alpha$, where T_{IHC} is the execution time of the original IHC algorithm. The interested reader will note that in the modified IHC algorithm, the outermost **for** loop must be changed to iterate from $i = \eta - 1$ down to $i = 0$. For simplicity, the original IHC algorithm is used in our analysis.

In a dedicated network, the IHC algorithm can be executed with $\eta = \mu$. Using $\mu = 2$, we get the following comparisons on execution time.

TABLE IV
WORST-CASE EXECUTION TIMES

Algorithm	Execution Time
IHC	$\eta(N-1)(\tau_S + \mu\alpha + D)$
VRS-ATA	$N(\log_2 N + 1)(\tau_S + \mu\alpha + D)$
KS-ATA	$N(2\sqrt{(N-1)/3} - 2)(\tau_S + \mu\alpha + D)$
VSQ-ATA	$N(2\sqrt{N} - 3)(\tau_S + \mu\alpha + D)$
FRS [12]	$(\log_2 N + 1)(\tau_S + D) + (N-1)\mu\alpha$

In Table III, the IHC algorithm is clearly better than all of the other algorithms. In [8], Dally quoted a time of 20 ns for routing a single packet through a node using cut-through. Using this figure, the time required for ATA reliable broadcast using the IHC algorithm is $2\tau_S + 0.02$ ms on a 1024-node Q_{10} and $2\tau_S + 1.31$ ms on a 64 K-node Q_{16}. (Using a conservative figure of $\tau_S = 0.5$ ms, this implies that over 68.7 billion packets ($\gamma N(N-1)$) can be sent and received in 1.81 ms on a 64 K-node hypercube.) It is feasible to dedicate the interconnection network to the ATA reliable broadcast operation for this length of time. If several communication channels run through each link, it is also feasible to dedicate one channel on each link to the ATA reliable broadcast operation for this length of time.

B. Worst-Case Analysis

The worst-case performance for the IHC algorithm occurs when *all* of the potential cut-through operations have to be performed as store-and-forward operations due to heavy network traffic. Adding in the additional queueing delay D, the worst-case execution time for the IHC algorithm can be written as $\eta(N-1)(\tau_S + \mu\alpha + D)$. The worst-case execution times for the VRS-ATA, KS-ATA, and VSQ-ATA change in a similar manner to reflect the fact that *all* send-receive operations have to be buffered and experience the queueing delay D. The worst-case execution time for the FRS [12] algorithm is the same as in Table II with the addition of the queueing delay D to the startup delay τ_S. Table IV summarizes the worst-case execution times of the various ATA reliable broadcast algorithms considered.

From Table IV, it can easily be seen that the FRS algorithm has the best performance as $(\log_2 N + 1)$ is multiplied to $\tau_S + D$ as opposed to a factor of N for the other algorithms. The worst-case performance of the cut-through algorithms results when *none* of the *potential* cut-through operations can be implemented with cut-through—this happens with extremely high network traffic. In this case, since all send-receive operations must experience the queueing delay D and startup delay τ_S, performance degradation is minimized if packets from individual nodes are merged into larger messages before being sent on the next step; since the FRS algorithm performs this operation in a highly efficient manner, it has the best performance given extremely high network traffic (large ρ). In the general case of $\rho > 0$, the execution times of the various algorithms will fall between the best and worst-case execution times shown in Tables II and IV, respectively.

Tables II and IV provide comparisons of various possible ATA reliable broadcast algorithms given highly favorable and highly unfavorable network conditions, respectively. Table II

and Theorem 4 show that the IHC algorithm is the "best possible" algorithm in terms of execution time when the network is under-loaded. Table IV shows that a store-and-forward algorithm such as the FRS algorithm is the "best possible" when the network is heavily-loaded. While this analysis can serve as a basis for comparison with other ATA reliable broadcast algorithms, a practical implementations of the IHC algorithm will require the solution of several practical issues such as the packet format, timing message reconstruction, and control.

VII. Conclusion

The all-to-all (ATA) reliable broadcast problem has been considered for a class of regular interconnection networks. Although good solutions to this problem exist if we are restricted to using store-and-forward switching, better solutions can be found by taking advantage of the faster communication possible with cut-through switching. When designing an ATA reliable broadcast algorithm that uses cut-through, we would like to ensure that all possible cut-throughs can be achieved when there is no other network traffic. This implies that nodes must be at least loosely synchronized and coordinate their actions such that "collisions" of broadcast packets do not occur. One method of doing this to stagger the nodes that initiate reliable broadcasts and to finish the individual reliable broadcasts as fast as possible. Another method is to interleave the nodes that initiate reliable broadcasts. However, with this latter method, the reliable broadcasts by individual nodes should be performed such that two broadcast packets from the same or different nodes do not interfere with each other's progress.

In this paper, we have proposed a general method for performing ATA reliable broadcast in an interleaved manner. Our solution, the IHC algorithm, compares favorably to several other possible cut-through and store-and-forward algorithms. Unlike the other algorithms studied, the IHC algorithm can be used on a large class of interconnection networks. In terms of execution time, the IHC algorithm with the minimum value of η is shown to be optimal. Moreover, using a cut-through time of 20 ns, the IHC algorithm can complete the ATA reliable broadcast in a few milliseconds even on a 64 K-node system. Thus, it is feasible to dedicate the entire network (or one channel on each directed link) to the IHC algorithm for the duration of the broadcast. In heavy network traffic, a store-and-forward algorithm that merges packets as they are being relayed has the best overall performance. However, among the possible cut-through algorithms considered, the IHC algorithm still has the best worst-case performance.

Appendix
List of Symbols and Acronyms

ATA	all-to-all
α	time for a broadcast packet to cut through an intermediate node
B_{FIFO}	size of the FIFO buffer at the receivers
D	queueing delay experienced by a buffered broadcast packet
η	interleaving distance used in the IHC algorithm
FRS [10]	Fraigniaud's all-to-all reliable broadcast algorithm
G^{dir}	a graph G with every undirected edge replaced by two directed edges
γ	connectivity of G
HARTS	hexagonal architecture for real-time systems
HC	Hamiltonian cycle
HC_j^i-cycle	transmission of packets in cycle HC_j during stage i of IHC
H_m	C-type wrapped hexagonal mesh of size m
IHC	Proposed ATA reliable broadcast algorithm
Λ	special class of regular interconnection networks defined in Section II
KS [15]	Kandlur and Shin's reliable broadcast algorithm for hex-meshes
KS-ATA	ATA reliable broadcast algorithm based on the KS [15] algorithm
μ	length of a broadcast packet divided by B_{FIFO}
N	number of nodes in the network
Q_m	hypercube of dimension m
RS [20]	Ramanathan and Shin's reliable broadcast algorithm for hypercubes
ρ	utilization of available communication links by normal system tasks
SQ_m	$m \times m$ torus-wrapped square mesh
τ_S	message startup time for store-and-forward
VRS	RS algorithm modified to use virtual cut-through
VRS-ATA	Ata reliable broadcast algorithm based on the VRS algorithm
VSQ	virtual cut-through reliable broadcast algorithm for a SQ_m
VSQ-ATA	ATA reliable broadcast algorithm based on the VSQ algorithm

Acknowledgment

The authors would like to thank D. D. Kandlur, J. W. Dolter, and other members of the Real-Time Computing Laboratory for helpful discussions on the ideas presented in this paper. The proof of Theorem 1 was based on a personal communique received from C. T. Ho.

References

[1] B. Alspach, J. C. Bermond, and D. Sotteau, "Decomposition into cycles 1: Hamiltonian decompositions," in *Cycles and Rays*, G. Hahn, G. Sabidussi, and R.E. Woodrow, Eds. Boston: Kluwer Academic, 1990, pp. 9–18.

[2] J. Arbert and B. Schneider, "Decomposition de la somme cartesienne d'un cycle et de l'union de deux cycles Hamiltoniens en cycles Hamiltoniens," *Discrete Math.*, vol. 38, pp. 7–20, 1982.

[3] B. Becker and H. U. Simon, "How robust is the n-cube?," in *Proc. 27th Annu. Symp. on Found. of Comput. Sci.*, Oct. 1986, pp. 283–291.

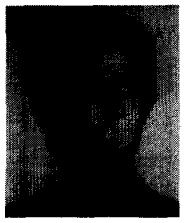
[4] J. A. Bondy and U. S. R. Murty, *Graph Theory with Applications*. New York: North-Holland, 1976.

[5] M. S. Chen, K. G. Shin, and D. D. Kandlur, "Addressing, routing and broadcasting in hexagonal mesh multiprocessors," *IEEE Trans. Comput.*, vol. C-39, no. 1, pp. 10–18, Jan. 1990.

[6] W. J. Dally and C. L. Seitz, "The torus routing chip," *J. Distribut. Computing*, vol. 1, no. 3, pp. 187–196, 1986.

[7] ——, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Trans. Comput.*, vol. C-36, no. 5, pp. 547–553, May 1987.

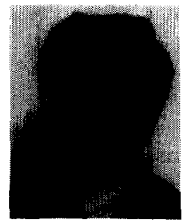
[8] W. J. Dally, "A fine-grain, message-passing processing node," in *Concurrent Computations: Algorithms, Architecture, and Technology*. New York: Plenum Press, 1988, pp. 375–389.
[9] D. Dolev, "The Byzantine generals strike again," *J. Algorithms*, vol. 3, pp. 14–30, 1982.
[10] J. W. Dolter, P. Ramanathan, and K. G. Shin, "A microprogrammable VLSI routing controller for Harts," in *Proc. ICCD'89*, Oct. 1989, pp. 160–163.
[11] M. Foregger, "Hamiltonian decompositions of products of cycles," *Discrete Math.*, vol. 24, pp. 251–260, 1978.
[12] P. Fraigniaud, "Asymptotically optimal broadcast and total-exchange algorithms in faulty hypercube multicomputers," Laboratories de l'Informatique du Parallelisme, Ecole Normal Superieure de Lyon, May 1989.
[13] C. T. Ho, personal communication, Nov. 1989.
[14] H. Kanakia and D. R. Cheriton, "The VMP network adapter broad (NAB): High-performance network communication for multiprocessor," in *SIGCOMM '88*, Aug.t 1988, pp. 175–187.
[15] D. D. Kandlur and K. G. Shin, "Reliable broadcast in hexagonal mesh multiprocessors," *ACM Trans. Comput. Syst.*, vol. 9, no. 4, pp. 374–398, Nov. 1991.
[16] P. Kermani and L. Kleinrock, "Virtual cut-through: A new computer communication switching technique," *Comput. Netw.*, vol. 3, no. 4, pp. 267–286, Sept. 1979.
[17] C. M. Krishna, K. G. Shin, and R. W. Butler, "Ensuring fault tolerance of phase-locked clocks," *IEEE Trans. Comput.*, vol. C-34, no. 8, pp. 752–756, Aug. 1985.
[18] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," *ACM Trans. Prog. Languages and Syst.*, vol. 4, no. 3, pp. 382–401, July 1982.
[19] L. Lamport and P. M. Melliar-Smith, "Synchronizing clocks in the presence of Faults," *J. ACM*, vol. 32, no. 1, pp. 52–78, January 1985.
[20] P. Ramanathan and K. G. Shin, "Reliable broadcast in hypercube multicomputers," *IEEE Trans. Comput.*, vol. 37, no. 12, pp. 1654–1657, Dec. 1988.
[21] P. Ramanathan, D. D. Kandlur, and K. G. Shin, "Hardware-assisted software clock synchronization for homogeneous distributed systems," *IEEE Trans. Comput.*, vol. 39, no. 4, pp. 514–524, Apr. 1990.
[22] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public key cryptosystems," *Commun. ACM*, vol. 21, pp. 120–126, Feb. 1978.
[23] C. L. Seitz, "The cosmic cube," *Commun. ACM*, vol. 28, pp. 22–33, Jan. 1985.
[24] H. Sullivan, T. Bashkov, and D. Klappholz, "A large scale, homogeneous, fully distributed parallel machine," in *Proc. Fourth Annu. Symp. Comput. Architecture*, March 1977, pp. 105–124.
[25] C. L. Yang and G. M. Masson, "A distributed algorithm for fault diagnosis in systems with soft failures," *IEEE Trans. Comput.*, vol. 37, no. 11, pp. 1476–1479, Nov. 1988.



Sunggu Lee (S'87–M'90) received the B.S.E.E. degree (with highest distinction) from the University of Kansas, Lawrence, in 1985, and the M.S.E.E. and Ph.D. degrees from the University of Michigan, Ann Arbor, in 1987 and 1990, respectively.

He is currently an Assistant Professor in the Department of Electronic and Electrical Engineering at the Pohang University, Pohang, Republic of Korea. Prior to this appointment, he was an Assistant Professor in the Department of Electrical Engineering at the University of Delaware, Newark, DE. His research interests are in parallel and fault-tolerant computing.

Dr. Lee is a member of the IEEE Computer Society and the Tau Beta Pi Engineering Honor Society.



Kang G. Shin (S'75–M'78–SM'83–F'92) received the B.S. degree in electronics engineering from Seoul National University, Seoul, Republic of Korea, in 1970, and the M.S. and Ph.D. degrees in electrical engineering from Cornell University, Ithaca, NY, in 1976 and 1978, respectively.

He is Professor and Director of the Real-Time Computing Laboratory, Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor. From 1978 to 1982, he was on the faculty of Rensselaer Polytechnic Institute, Troy, NY. He has held visiting positions at the U.S. Air Force Flight Dynamics Laboratory, AT&T Bell Laboratories, the computer science division within the Department of Electrical Engineering and Computer Science, University of California, Berkeley, and the International Computer Science Institute, Berkeley, CA. He also chaired the computer science and engineering division, Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, for three years beginning in 1991. In 1985, he founded the Real-Time Computing Laboratory, where he and his colleagues are currently building a 19-node hexagonal mesh multicomputer, called **HARTS**, to validate various architectures and analytic results in the area of distributed real-time computing. He has also been applying the basic research results of real-time computing to manufacturing-related applications ranging from the control of robots and machine tools to the development of open architectures for manufacturing equipment and processes. Recently, he has initiated research on the open-architecture information base for machine tool controllers.

Dr. Shin has authored or co-authored over 270 technical papers (more than 120 of these in archival journals) and several book chapters in the areas of distributed real-time computing and control, fault-tolerant computing, computer architecture, robotics and automation, and intelligent manufacturing. In 1987, he received the Outstanding IEEE TRANSACTIONS ON AUTOMATIC CONTROL Paper Award for a paper on robot trajectory planning. In 1989, he also received the Research Excellence Award from the University of Michigan. He was the Program Chairman of the 1986 IEEE Real-Time Systems Symposium (RTSS), the General Chairman of the 1987 RTSS, the Guest Editor of the 1987 August special issue of IEEE TRANSACTIONS ON COMPUTERS on real-time systems, a Program Co-Chair for the 1992 International Conference on Parallel Processing, and served numerous technical program committees. He also chaired the IEEE Technical Committeeon Real-Time Systems during 1991–1993. He is also a Distinguished Visitor of the IEEE Computer Society, an Editor of IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED COMPUTING, and an Area Editor of *International Journal of Time-Critical Computing Systems*.