Addressing, Routing, and Broadcasting in Hexagonal Mesh Multiprocessors

MING-SYAN CHEN, MEMBER, IEEE, KANG G. SHIN, SENIOR MEMBER, IEEE, AND DILIP D. KANDLUR

***Abstract*—A family of 6-regular graphs, called *hexagonal meshes* or *H-meshes*, is considered as a multiprocessor interconnection network. Processing nodes on the periphery of an *H*-mesh are first wrapped around to achieve regularity and homogeneity. The diameter of a wrapped *H*-mesh is shown to be of $O(p^{1/2})$, where p is the number of nodes in the *H*-mesh. An elegant, distributed routing scheme is developed for wrapped *H*-meshes so that each node in an *H*-mesh can compute shortest paths from itself to any other node with a straightforward algorithm of $O(1)$ using the addresses of the source-destination pair only, i.e., independent of the network's size. This is in sharp contrast with those previously known algorithms that rely on using routing tables. Furthermore, we also develop an efficient point-to-point broadcasting algorithm for the *H*-meshes which is proved to be optimal in the number of required communication steps.**

The wrapped *H*-meshes are compared against some other existing multiprocessor interconnection networks, such as hypercubes, trees, and square meshes. The comparison reinforces the attractiveness of the *H*-mesh architecture.

***Index Terms*—Addressing and routing strategies, hexagonal mesh, interconnection networks, node homogeneity and regularity, regular graphs.**

I. Introduction

MULTIPROCESSOR interconnection networks are often required to connect thousands of homogeneously replicated processor-memory pairs [1], [2], each of which is called a *processing node* (PN). Instead of using a shared memory, all synchronization and communication between PN's for program execution is often done via message passing. Design and use of multiprocessor interconnection networks have recently drawn considerable attention due to the availability of inexpensive, powerful microprocessors and memory chips [3]-[6]. The homogeneity of PN's and the interconnection network is very important because it allows for cost/performance benefits from the inexpensive replication of multiprocessor components. (See [7]-[9] for more justifications.) Each PN in the multiprocessor is preferred to have fixed connectivity so that standard VLSI chips can be used. Also, the interconnection network needs to contain a reasonably high degree of redundancy so that alternative routes can be made available to detour faulty nodes/links. More importantly, the interconnection network must facilitate efficient routing and broadcasting so as to achieve high performance in job executions.

Various research and development results on how to interconnect multiprocessor components have been reported in the literature [10], [11], [9], [12], [13]. Some of them have been compared to each other [3], [14]. However, most of them are not satisfactory mainly due to their inability of providing *all* the following features: the simplicity of interconnection, the efficiency of message routing and broadcasting, a fine scalability,[1] and a high degree of fault tolerance.

In this paper, hexagonal meshes (or *H*-meshes) are presented as a candidate multiprocessor architecture that possesses all the foregoing salient features. A large number of data manipulation applications require the PN's on the hexagonal periphery to be wrapped around to achieve regularity and homogeneity such that identical software and protocols can be applied uniformly over the network. From a topological point of view, the way of wrapping determines the type of *H*-mesh. For example, Martin proposed a general method for forming a "boundary-less" topology for a large, dense, and regular arrangement of processor and memory modules on a two-dimensional surface, called a *processing surface* [15]. He showed specifically how to form such a topology on a square mesh and implied, without any elaboration, that a similar approach could be applied to different processing surfaces. Stevens illustrated an interesting way of wrapping *H*-meshes of size less than or equal to three, where the size of an *H*-mesh is defined as the number of nodes on each peripheral edge of the *H*-mesh [9]. His method for addressing, message routing, and broadcasting in *H*-meshes, however, is very complex and inefficient.

To the best of our knowledge, there is no general efficient method for wrapping, routing, and broadcasting in *H*-meshes known to date. Consequently, in this paper, a method for systematically wrapping *H*-meshes of arbitrary size is presented, and a new, simple addressing scheme for the *H*-meshes is proposed. Then, efficient routing and broadcasting algorithms under the new addressing scheme are developed. As we shall see, the proposed addressing scheme not only achieves the homogeneity of PN's but also facilitates routing and broadcasting in the *H*-meshes significantly.

The rest of this paper is organized as follows. In Section

Manuscript received June 15, 1987; revised December 30, 1987. This work was supported in part by the Office of Naval Research under Contracts N00014-87-G-0086 and N00014-85-K-0122. Any opinions, findings, and recommendations expressed in this paper are those of the authors and do not necessarily reflect the view of the ONR.

M.-S. Chen is with IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598.

K. G. Shin and D. D. Kandlur are with the Real-Time Computing Laboratory, Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor, MI 48109.

IEEE Log Number 8931922.

[1]Scalability is measured in terms of the number of PN's necessary to increase the network's dimension by one.

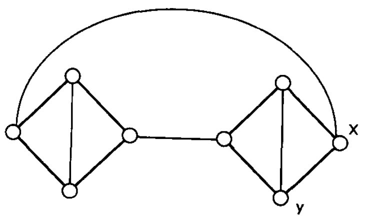


Fig. 1. A regular nonhomogeneous graph.

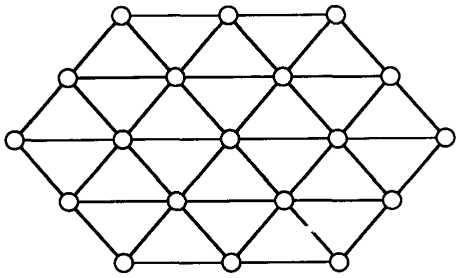


Fig. 2. An H-mesh of size 3 without wrapping.

II, a systematic way to wrap the H-mesh, called the *continuous type* (*C-type*) wrapping, is presented as a means of achieving the regularity and the homogeneity of PN's. Topological properties of H-meshes are explored in Section III. In light of these properties, an addressing scheme for a C-type wrapped H-mesh is proposed in Section IV. With that addressing scheme, efficient routing and broadcasting algorithms are then developed. In Section V, we compare the C-type wrapped H-meshes with some other existing multiprocessor structures, such as hypercubes, trees, and square meshes. The paper concludes with Section VI.

II. Systematic Wrapping of H-Meshes

A graph is said to be *regular* if all the nodes in the graph have the same degree, and *homogeneous* if all the nodes in that graph are topologically identical. Clearly, homogeneity implies regularity, but the converse does not always hold. For example, the graph in Fig. 1 is regular, but not homogeneous since node x and node y are not topologically identical. The degree of all nodes in an H-mesh without wrapping, except those on its periphery, is 6. Thus, such an H-mesh is neither homogeneous nor regular. Fig. 2 illustrates an unwrapped H-mesh of size 3.

Consider the central node of the H-mesh in Fig. 3. The node has six oriented directions, each of which leads to one of its six nearest neighbors. Without loss of generality, any of the six directions can be defined as the *x direction*, the direction 60 degrees clockwise to the x direction as the *y direction*, and then the direction 60 degrees clockwise to the y direction as the *z direction*. Once the x, y, and z directions are defined, an H-mesh of size n can be partitioned into $2n-1$ rows with respect to *any* of these three directions; there are three different ways of partitioning the rows of an H-mesh. Fig. 3 shows the rows in an H-mesh of size 3 which are partitioned with respect to the x, y, and z directions, respectively. To facilitate our presentation, when an H-mesh of size n is partitioned into $2n-1$ rows with respect to any of the three

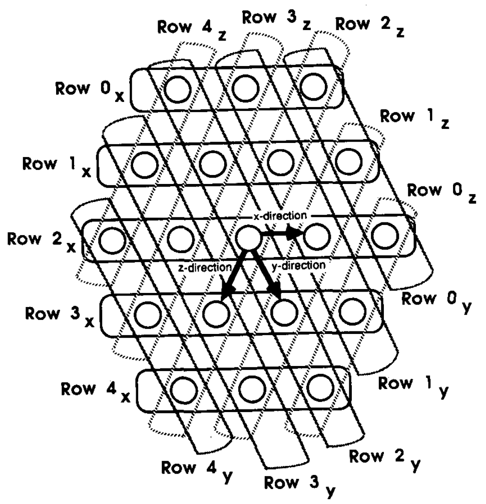


Fig. 3. Partitioned rows of an H_3 in x, y, and z directions.

directions, and the H-mesh is rotated in such a way that the corresponding direction from the central node points to the right, the top row is referred to as row 0 in that direction, the second to the top row is referred to as row 1, and so on. Also, row $n-1$ is called the *central* row and rows 0 to $n-2$ and rows n to $2n-2$ will sometimes be referred to as the *upper* and *lower* parts of an H-mesh of size n, respectively. (Subscripts for the row numbers in Fig. 3 are used to indicate their directions.)

For notational simplicity, let $[b]_a \equiv b \bmod a$, for all $a \in \boldsymbol{I}^+$ and $b \in \boldsymbol{I}$, where $\boldsymbol{I}$ is the set of integers and $\boldsymbol{I}^+$ the set of positive integers. To make the H-mesh homogeneous and regular, the following method for wrapping an H-mesh continuously, called the *C-type wrapping*, is proposed.

C-type wrapping: Wrap an H-mesh of size n in such a way that for each of the three ways of partitioning the PN's into rows, the last PN in the row i is connected to the first PN of row $[i+n-1]_{2n-1}$.

As will be stated later in Corollary 1.1, the PN's in an H-mesh with the C-type wrapping are homogeneous. Furthermore, the C-type wrapping allows for an easy addressing scheme and, thus, simplifies the routing and broadcasting in an H-mesh significantly. In what follows, an H-mesh of size n with the C-type wrapping is denoted by H_n, while that without wrapping is denoted by H'_n.

The edges in the rows partitioned with respect to the x (y or z) direction and the associated wrapping are called the *edges in the x (y or z) direction*. An illustrative example of partitioning edges in an H_3 into three different directions is given in Fig. 4, where the edges in each direction are drawn separately for clarity. From Figs. 3 and 4, it can be observed that there is a unique path from one node to another in an H_n along each of the three directions.

III. Topological Properties of Hexagonal Meshes

The following two lemmas are direct results of the structure of an H-mesh.

Lemma 1: The number of nodes in an H_n is $p = 3n^2 - 3n + 1$.

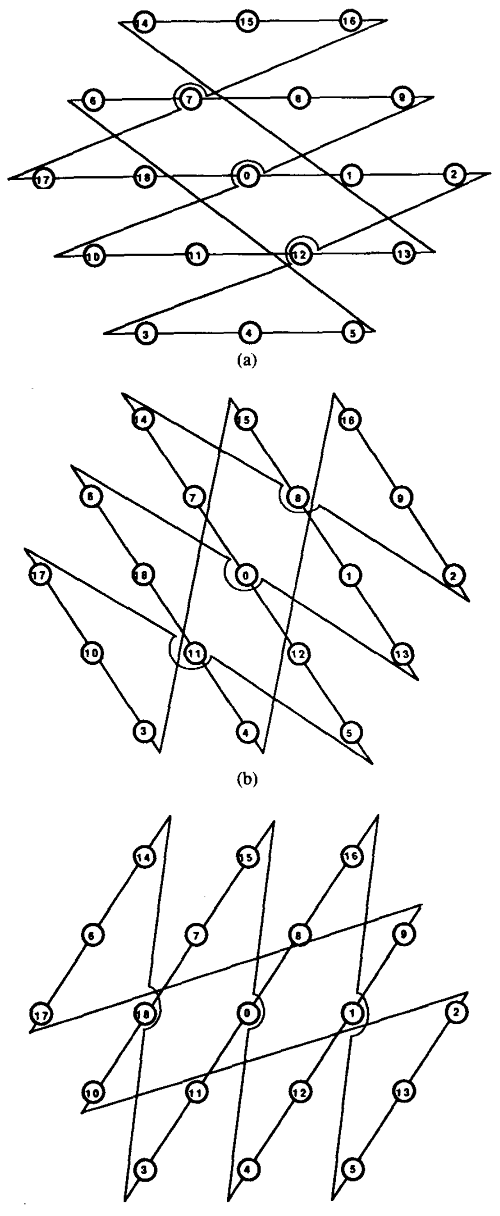


Fig. 4. An H_3 with the C-type wrapping. (a) Edges in the x-direction. (b) Edges in the y-direction. (c) Edges in the z- direction.

Proof: Since there are always $n+i$ nodes in row i as well as in row $2n-i-2$ for $0 \le i \le n-2$, and $2n-1$ nodes in row $n-1$, the desired result follows from $\sum_{i=0}^{n-2} 2(n+i)+2n-1 = 2n(n-1)+(n-1)(n-2)+2n-1 = 3n^2-3n+1$. Q.E.D.

Lemma 2: The sum of the number of nodes in row i and that of row $i+n$ is $3n-2$ for $0 \le i \le n-2$.

The proof of Lemma 2 is trivial and, thus, omitted.

To exploit the topological properties of an H_n, each node is labeled with a 3-tuple as follows. Start from the central node of the H_n and label that node with $(0,0,0)$, where the first coordinate of a node is referred to as its x labeling and

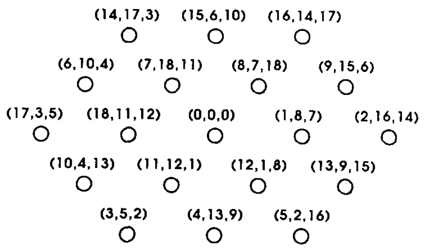


Fig. 5. An H_3 with a 3-tuple labeling.

the second and third coordinates are referred to as its y and z labelings, respectively. Then, move to the next node along the x direction, assign that node the x labeling one more than that of its preceding node, and so on. The y and z labelings for each node are determined by moving along the y and z directions, respectively, instead of the x direction. An example of the 3-tuple labeling of an H_3 is given in Fig. 5, where the edges are not drawn for clarity.

Theorem 1: Let (x_1, y_1, z_1) and (x_2, y_2, z_2) be, respectively, the labelings of nodes n_1 and n_2 in an H_n and $p = 3n^2 - 3n + 1$. Then,

a) $[x_2 - x_1]_p = [(3n^2 - 6n + 3)(y_2 - y_1)]_p$.

b) $[x_2 - x_1]_p = [(3n^2 - 6n + 2)(z_2 - z_1)]_p$.

Proof: For any pair of adjacent nodes n_u and n_v in the y-direction, respectively, with the labelings (x_u, y, z_u) and $(x_v, [y+1]_p, z_v)$, we want to claim $[x_v - x_u]_p = 3n^2-6n+3$. We shall prove this claim first and then a) follows immediately from the claim.

Consider a path P^* from n_u to n_v following the $+x$ direction *only*. (As can be seen in Fig. 4(a), such a path is unique.) Suppose n_u is in row r_u according to the row partition with respect to the x direction. Recall that the C-type wrapping requires the end of a row to be connected to the beginning of a row that is $n-1$ rows away from it. Since $[(n-1)(2n-3)]_{2n-1} = 1$, P^* must run through nodes in $2n-2$ different rows to get from n_u to n_v—which are adjacent in the y direction—including the rows that contain n_u and n_v. In other words, the path P^* must visit all but

1) those nodes ahead of n_u in row r_u,
2) those nodes behind n_v in row $[r_u+1]_{2n-1}$, and
3) those nodes in row $[r_u+n]_{2n-1}$ (the only row that P^* does not travel).

Note that if n_u is in the upper part or central row of an H_n, the total number of nodes in 1) and 2) will be one less than the number of nodes in row r_u. On the other hand, if n_u is in the lower part of the H_n, the total number of nodes in 1) and 2) will be one less than the number of nodes in row $[r_u+1]_{2n-1}$. By Lemma 2, we conclude that for both cases, the total number of nodes in 1), 2), and 3) is $3n-3$, i.e., one less than $3n-2$. Thus, the number of nodes on P^* is $(3n^2-3n+1)-(3n-3) = 3n^2-6n+4$. Since both n_u and n_v are contained in P^*, the claim is thus proved and a) follows.

For b), we claim that for any pair of adjacent nodes n_u and n_v in the z direction labeled, respectively, with (x_u, y_u, z) and $(x_v, y_v, [z+1]_p)$, $[x_v - x_u]_p = 3n^2-6n+2$. This claim can be proved by following the same logic as above and, thus, b) can be proved similarly. Q.E.D.

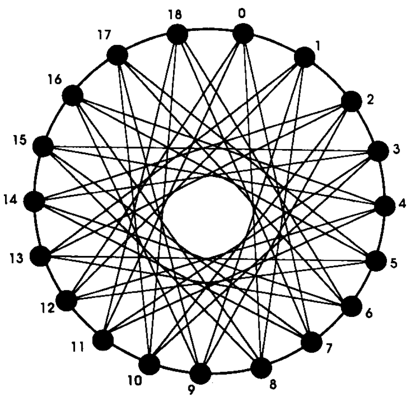


Fig. 6. A redrawn H_3.

Note that $(3n^2 - 3n + 1) - (3n^2 - 6n + 2) = 3n - 1$, $(3n^2 - 3n + 1) - (3n^2 - 6n + 3) = 3n - 2$. In light of Theorem 1, an H_n can be redrawn as a power cycle with $p = 3n^2 - 3n + 1$ nodes, in which node i is not only adjacent to nodes $[i-1]_p$ and $[i+1]_p$, but also adjacent to nodes $[i+3n-1]_p$, $[i+3n-2]_p$, $[i+3n^2-6n+2]_p$, and $[i+3n^2-6n+3]_p$. For example, an H_3 can be redrawn as the one in Fig. 6. This fact leads to the following corollary.

Corollary 1.1: All the processing nodes in an H_n are homogeneous.

Note that, although extensive results have been obtained for many classes of networks [16]–[19], the connection pattern in an H-mesh does not belong to any of the previously found patterns.

Lemma 3:

a) The number of links in an H'_n is $9n^2 - 15n + 6$.

b) The number of links in an H_n is $9n^2 - 9n + 3$.

Proof: We prove b) first. From the fact that the summation of all node degrees in a graph is twice the number of edges in that graph, we have $6(3n^2-3n+1)/2 = 9n^2-9n+3$.

Since there are $6(n-1)$ nodes in the periphery of an H'_n, six of which have degree 3 and $6n-12$ of which have degree 4, we have the summation of all node degrees in an H'_n: $6(3n^2-3n+1-6n+6)+6\times3+(6n-12)4 = 18n^2-30n+12$, making the number of links in an H'_n equal to $9n^2 - 15n + 6$. Q.E.D.

Lemma 4:

a) The diameter of an H_n is $n - 1$.

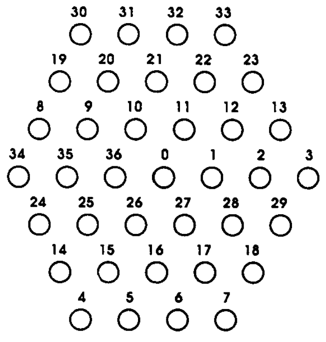
b) The average distance in an H_n is $(2n-1)/3$.

Proof: a) follows from the fact that, without loss of generality, every node in a wrapped H-mesh can view itself as the central node of the mesh. To prove b), let $\mathrm{td}(H_n)$ denote the summation of distances from any node to all the other nodes in an H_n. Then,

$$\mathrm{td}(H_n) = \sum_{d=1}^{n-1} 6d^2 = 6\frac{n(n-1)(2n-1)}{6} = n(n-1)(2n-1).$$

The average distance is thus $\mathrm{td}(H_n)/(p-1) = (2n-1)/3$. Q.E.D.

Moreover, Lemma 1 and a) of Lemma 4 lead to the following lemma.



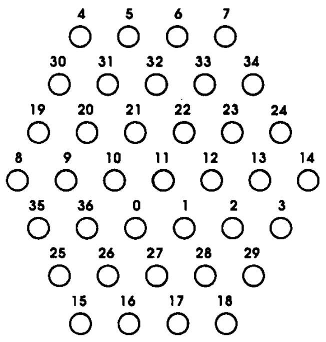


Fig. 7. An illustrative example of Theorem 2. (a) An H_4. (b) An $H_{4(11)}$.

Lemma 5: The diameter of an H-mesh with p nodes grows as $O(p^{1/2})$.

In addition, it is worth mentioning that H-meshes possess a high degree of fault tolerance measured in terms of *connectivity*. Recall that a graph is said to be *m-connected* (*m-edge connected*) if m is the minimal number of nodes (edges) whose removal will disconnect the graph [5]. It can be easily verified that an H-mesh of any size is 6-connected and also 6-edge connected. This means that an H-mesh can tolerate up to five node/link failures at a time, which is better than most of the other existing networks [14], [20].

IV. Routing and Broadcasting in H-Meshes

In this section, a simple addressing scheme for H-meshes is developed on the basis of the C-type wrapping. Under this addressing scheme, shortest paths from one node to any other node can be computed by the source node with an extremely simple algorithm of $O(1)$. A point-to-point broadcasting algorithm is also developed, which is proved to be optimal in the number of required communication steps.

A. Addressing Scheme

Using Theorem 1, the y and z labels of any node can be obtained from its x label, meaning that only one (instead of three) label will suffice to uniquely identify any node in an H-mesh. An example of this addressing for an H_4 is given in Fig. 7(a) where all edges are omitted for clarity. This addressing scheme is much simpler than the one in [9], since only one number, instead of two, is needed to identify each node in an H_n. Furthermore, the routing strategy under this addressing scheme will be shown to be far more efficient than the one in [9], especially when messages must be routed via wrapped links.

B. Routing

Under the above addressing scheme, a shortest path between any two nodes can be easily determined from the difference of their addresses.

Let m_x, m_y, and m_z be, respectively, the number of moves or hops from the source node to the destination node along the x, y, and z directions on a shortest path. Negative values mean the moves are in opposite directions. Note that there could be several equally short paths from a node to another, and these shortest paths are completely specified by the values of m_x, m_y, and m_z. More precisely, it can be verified that for $i = x, y, z$, the number of paths with m_i moves in the corresponding directions is $(|m_x|+|m_y|+|m_z|)!/(|m_x|!|m_y|!|m_z|!)$. Let s and d be, respectively, the addresses of the source and destination nodes, and $k = [d-s]_p$ where $p = 3n^2-3n+1$. Then, the m_x, m_y, and m_z for shortest paths from s to d can be determined by the algorithm given below.

Algorithm A_1:

```
begin
  $m_x := 0$; $m_y := 0$; $m_z := 0$;
  if ($k < n$) then begin $m_x := k$; stop end;
  if ($k > 3n^2-4n+1$) then begin $m_x := k-3n^2+3n-1$;
  stop end;
  $r := (k-n)$ div $(3n-2)$;
  $t := (k-n)$ mod $(3n-2)$;
  if ($t \le n+r-1$)
    then /* $d$ is in the lower part of the $H$-mesh centered at
      $s$. */
      if ($t \le r$)
        then $m_x := t-r$; $m_z := n-r-1$;
        else if ($t \ge n-1$) then $m_x := t-n+1$; $m_y :=$
          $n-r-1$;
            else $m_y := t-r$;   $m_z := n-t-1$;
          endif;
      endif;
    else /* $d$ is in the upper part of the $H$-mesh centered at
        $s$. */
      if ($t \le 2n-2$)
        then $m_x := t+2-2n$; $m_y := -r-1$;
        else if ($t \ge 2n+r-1$) then $m_x := t-2n-r+1$;
          $m_z := -r-1$;
            else $m_y := t+1-2n-r$; $m_z := 2n-t-2$;
          endif;
      endif;
  endif;
  stop
end;
```

The correctness of A_1 is proved by the theorem below.

Theorem 2: The values of m_x, m_y, and m_z determined by A_1 completely specify all the shortest paths from s to d in an H_n.

Proof: By Theorem 1, without loss of generality the source node can view itself as the central node of the H_n. Let $H_{n(s)}$ be the H-mesh centered at s. For the case when d is in the central row (i.e., row $n-1$) of $H_{n(s)}$, m_x is determined by the statements in lines 2 and 3 of A_1, and $m_y = m_z = 0$.

Consider the case when d is in a row other than row $n-1$ of $H_{n(s)}$. Form a *group* of nodes with the nodes of rows $n-i-2$ and $2n-i-2$; call this *group i*. Then, in Fig. 3, rows 1 and 4 form group 0, and rows 0 and 3 form group 1. A group consists of two rows, one from the upper part and the other from the lower part of $H_{n(s)}$. We know from Lemma 2 that each group contains $3n-2$ nodes. Then, by the statements in lines 5 and 6 of A_1, r is determined as the identity of the group which contains d in $H_{n(s)}$, and t is the position of d in group r. We can determine from t which row of $H_{n(s)}$ contains d. Denote the first node of that row by n_f. A shortest path from s to n_f and that from n_f to d can thus be determined. Using the idea of the composition of vectors, we get the desired equations for m_x, m_y, and m_z. Q.E.D.

An illustrative example for routing in an H_4 is given in Fig. 7 where edges are omitted for clarity. Suppose node 11 sends a message to node 5, i.e., $n = 4$, $s = 11$, $d = 5$, and $k = [5-11]_{37} = 31$. The original H_4 is given in Fig. 7(a) and $H_{4(11)}$ is in Fig. 7(b), where node 11 is placed at the center of the H_4. (As mentioned before, this can be done without loss of generality.) From A_1, we get $r = 2$, $t = 7$ and then $m_x = 0$, $m_y = -2$ and $m_z = -1$. Note that the route from node 11 to node 5 is isomorphic to that from node 0 to node 31. This is not a coincidence, but rather, a consequence of the homogeneity of H_4. All paths from one node to another in an H_n are completely determined by the difference in their addresses.

Moreover, the complexity of A_1 is $O(1)$, which is independent of the size of H-mesh, and only the source node needs to execute the algorithm. Once m_x, m_y, and m_z are determined, they form a routing record to be sent along with a regular message. The routing in an H_n is then characterized by the following six routing functions:

$$R_{m_x-1}(i) = [i+1]_p$$

$$R_{m_x+1}(i) = [i-1]_p$$

$$R_{m_y-1}(i) = [i+3n^2-6n+3]_p$$

$$R_{m_y+1}(i) = [i+3n-2]_p$$

$$R_{m_z-1}(i) = [i+3n^2-6n+2]_p$$

$$R_{m_z+1}(i) = [i+3n-1]_p$$

where, as before, $p = 3n^2-3n+1$ is the total number of nodes in an H_n.

The routing record is updated by the above six functions at each intermediate node on the way to the destination node so that the current routing record may contain the correct information for the remaining part of the path. The above functions are applied repeatedly until $m_x = m_y = m_z = 0$, meaning that the message has reached the destination mode.

C. Point-to-Point Broadcasting

Applications in various domains require an efficient method for broadcasting messages to every node in an H-mesh. Due

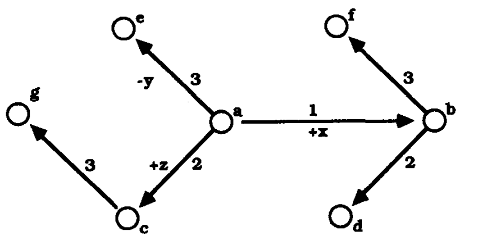


Fig. 8. First phase of the broadcasting algorithm.

to interconnection costs, it is very common to use point-to-point communications for broadcasting. In this subsection, we present a broadcasting algorithm using point-to-point communications. Furthermore, we shall prove that this algorithm is optimal in the number of required communication steps. We then apply this algorithm in developing a systematic procedure for computing the sum of numbers distributed across the nodes of an H-mesh.

Without loss of generality, we can assume the center node to be the source node of the broadcast. The set of nodes which have the same distance from the source node is called a *ring*. The main idea of our algorithm is to broadcast a message, ring by ring, toward the periphery of an H-mesh. The algorithm consists of two phases. In the first phase, which takes three steps, the message is transmitted to the six nearest neighbors of the origin. As shown in Fig. 8, node a sends the message along the $+x$ direction to node b in step 1, nodes a and b send messages along the z direction to nodes c and d, respectively, in step 2, and then, nodes a, b, and c send messages along the $-y$ direction to nodes e, f, and g, respectively, in step 3. At the end of this phase, nodes b, c, d, e, f, and g are assigned the directions x, z, y, $-y$, $-z$, and $-x$, respectively, as the propagation direction.

Note that there are six corner nodes in each ring. In the second phase, which takes $n-1$ steps, the six corner nodes of each ring send the message to two neighboring nodes, respectively, while all the other nodes propagate the message to the next node along the direction in which the message was previously sent. An illustrative example for the broadcasting in an H_4 is given in Fig. 9. A formal description of this procedure is given in algorithm A_2 below, where the five arguments in the *send* and *receive* commands denote, respectively, the message to be broadcast, the direction from which the message is received, the direction to propagate the message, a flag indicating whether it is a corner node or not, and the count of communication steps. Also, the function *rotate* is used for each corner node to determine the direction of the second transmission, in which the direction is rotated clockwise. For example, y = rotate(x, +60°), z = rotate(y, +60°), and $-x$ = rotate(z, +60°).

```
Algorithm A_2:
Procedure for the source node:
begin
  send(message, $x$, $x$, TRUE, 1);
  send(message, $z$, $z$, TRUE, 2);
  send(message, $-y$, -y, TRUE, 3);
end
```

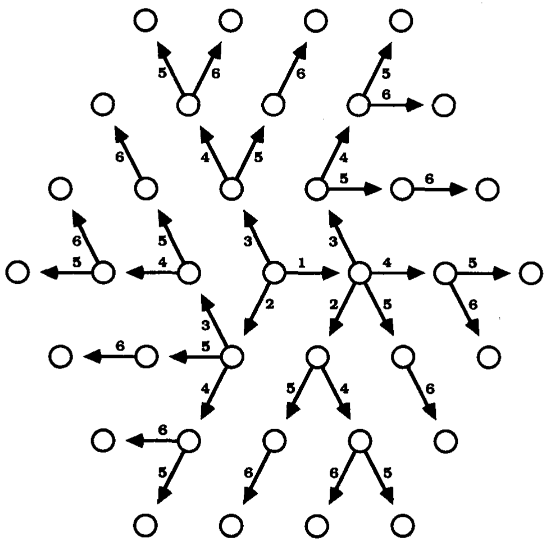


Fig. 9. Broadcasting in an H_4.

```
Procedure for all nodes except the source node:
begin
  receive(message, from-dirn, propagate-dirn, corner, count);
  case count of
  1: begin
       send(message, $z$, $y$, TRUE, 2);
       send(message, $-y$, $-z$, TRUE, 3);
       count := 3;
     end
  2: begin
       if (from-dirn = $z$) then send(message, $-y$, $-x$,
         TRUE, 3);
       count := 3;
     end
  $n+2$: stop        /* test for termination*/
  else:              /* do nothing*/
  endcase
  /* steps of the second phase */
  if (corner) and (count $\leq n+1$) then
  begin
    direction-2 := rotate (propagate-dirn, + 60°);
    send(message, propagate-dirn, propagate-dirn,
      TRUE, count + 1);
    send(message, direction-2, direction- 2, FALSE,
      count + 2);
  end
  else send(message, propagate-dirn, propagate-dirn,
    FALSE, count + 1);
end.
```

Note that the second phase is required only when $n > 2$. Thus, the total number of communication steps is $n+2$ when $n \geq 3$, and 3 when $n = 2$. Furthermore, we prove by the following theorem that A_2 is an optimal broadcasting algorithm in the number of required communication steps.

Theorem 3: Any broadcast algorithm for the hexagonal mesh H_n, $n \geq 3$, which uses point-to-point communication requires at least $n+2$ communication steps.

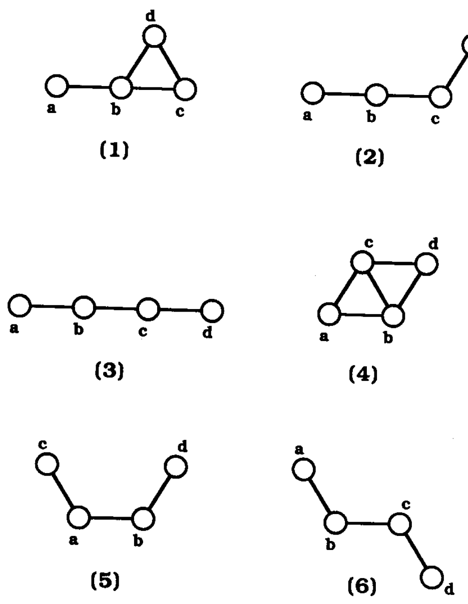


Fig. 10. Six possible patterns after two communication steps.

Proof: We first prove the result for H-meshes of size 3 and 4. An H_3 has 19 nodes and so, even if recursive doubling were used for each step of broadcasting, at least five steps are required to cover all the nodes. A similar argument applies for an H_4 which has 37 nodes, i.e., at least six steps are needed for the H_4. For larger values of n, we show that it is not possible to cover all nodes in an H_n in $n + 1$ steps.

To show this, we examine all possible patterns of nodes which can be reached using two communication steps. In two steps, only four nodes can be reached and when the duplicate patterns arising due to various symmetries are removed, we get only the six unique patterns shown in Fig. 10. These patterns can each be mapped onto the nodes on the periphery of an H_n, and in each case we can find at least five nodes which are $n - 1$ links away from each of the four mapped nodes. Note that, at most four nodes of these five can be reached in $n - 1$ steps from the original four nodes since only point-to-point communication is permitted. Hence, there is at least one node which cannot be reached in $n + 1$ steps. Therefore, at least $n + 2$ steps are required. Fig. 11 shows the mapping of the six possible patterns into an H_5, but it is clear from the figure that a similar mapping would apply for all larger meshes, and the theorem thus follows. Q.E.D.

To see an application of the broadcasting algorithm, consider the problem of computing the sum of numbers distributed across the nodes in the H-mesh. Note that this problem is important, since it occurs frequently in many applications such as the computation of an inner product of two vectors. The global sum can be obtained by first computing the inner product within each node for the segments of the vectors in the node and then summing these partial sums up. Typically, for vector operations, the inner product is required by all nodes. In light of the broadcasting algorithm, the procedure of obtaining the global sum can be accomplished systematically by $2n + 1$ communication steps as described below.

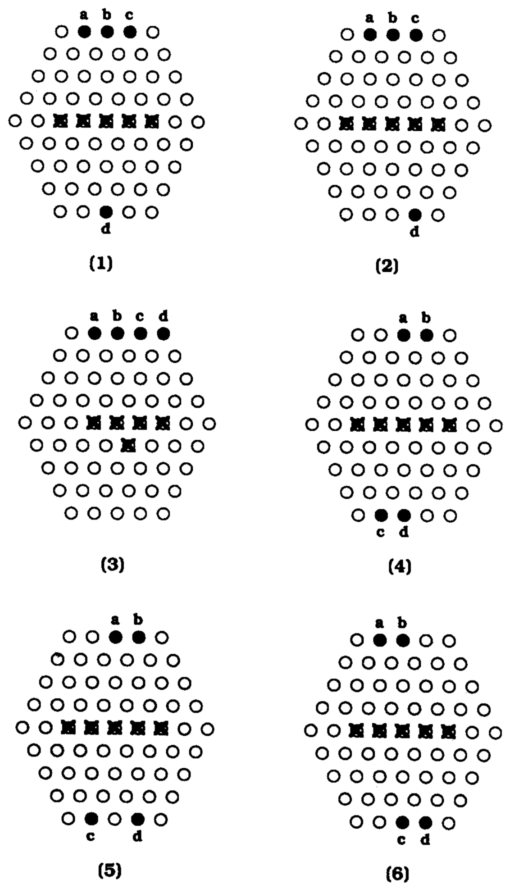


Fig. 11. Embedding in an H_5 of the six patterns in Fig. 10.

The procedure can be divided into two phases. In the first phase, the partial sums are transmitted toward a center node, while each node along the path computes the sum of all incoming partial sums and its own number, and then transmits the resulting partial sum inwards. An illustrative example of the first phase in an H_4 is shown in Fig. 12. It is easy to see that the first phase requires $n - 1$ communication steps in an H_n. In the second phase, the center node, after adding the six incoming partial sums with its own number, uses point-to-point broadcasting to distribute the sum to all nodes. Since we need $n + 2$ steps in the second phase, the total number of required communication steps for obtaining the global sum is $2n + 1$.

V. Comparative Remarks on H-Meshes

It is essential that messages in a large multiprocessor network be routed by each intermediate PN without using information about the entire network, since large storage and time overheads are required for maintaining such global information. As mentioned earlier, the C-type wrapping of an H-mesh not only provides regularity and homogeneity to the interconnection network but also allows for a very simple addressing scheme in the H-mesh, completely specifying all shortest paths from one node to any other node only with the ad-

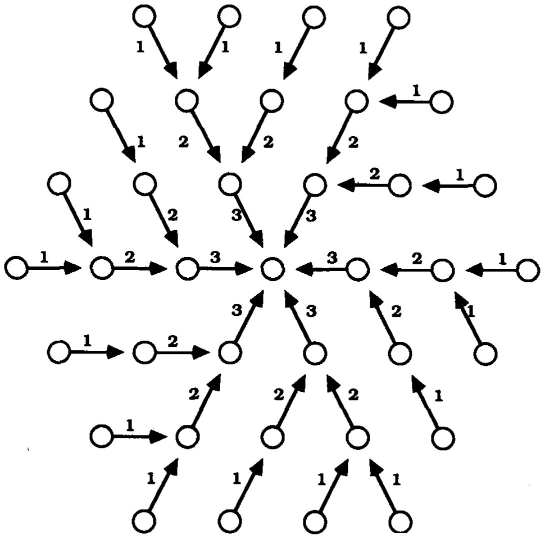


Fig. 12. First phase of the global sum algorithm in an H_4.

TABLE I
COMPARISONS AMONG VARIOUS MULTIPROCESSOR ARCHITECTURES

parameters / structures	node number	link number	diameter	node degree	connectivity	message density[†]
Complete graphs	p	p(p−1)/2	1	p − 1	p	$O(1/p)$
Hypercubes, Q_n	$p = 2^n$	$n2^{n-1}$	$O(\log_2 p)$	$\log_2 p$	$\log_2 p$	$O(1)$
k-level CBTs[††]	$p = 2^k - 1$	p − 1	$O(\log_2 p)$	root: 2 leaves: 1 others: 3	1	$O(\log_2 p)$
Stars	p	p − 1	2	center: p−1 branches: 1	1	center: p branches: 2
Cycles	p	p	$O(p)$	2	2	$O(p)$
Chordal rings	p	3p/2	$O(p^{1/2})$	3	3	$O(p^{1/2})$
Square meshes, w × w	$p = w^2$	2p	$O(p^{1/2})$	4	4	$O(p^{1/2})$
H-meshes, H_n	$p = 3n^2 - 3n + 1$	3p	$O(p^{1/2})$	6	6	$O(p^{1/2})$

[†]Message density is measured in messages per link per time interval under the assumption that each node sends a message to every other node within one time interval.

[††]CBT stands for a complete binary tree.

dresses of the source–destination pair. Shortest paths can then be computed by the source node with an elegant algorithm of complexity $O(1)$. This algorithm is better than those commonly used for hypercube multiprocessors that rely on an address comparison procedure [21], and those using routing tables of complexity $O(p)$, where p is the number of nodes in the system. Besides, in light of homogeneity, a systematic broadcasting algorithm, which has been proved to be optimal in the number of communication steps, is also available.

Several important features of various architectures are compared to those of H-meshes and tabulated in Table I, where connectivity is defined as the minimal number of nodes whose removal will disconnect the network and the O-notation is used to denote the complexity as a function of the network size. Although complete graphs and stars have constant diameters [22], they are not attractive because of the excessive number of connections required for complete graphs and the poor reliability of stars. On the other hand, the diameter of a hypercube is allowed to grow as $O(\log_2 p)$ at the cost of increasing the node degree, which could cause a serious wiring problem with the expansion of the network (i.e., fan-in and fan-out problems) and may make it difficult to implement with standard VLSI chips. This is viewed as a major drawback of the hypercube structure [1].

Consider the cases when both the hypercube and hexagonal mesh architectures have approximately the same number of nodes. A seven-dimensional hypercube, denoted by Q_7, has 128 nodes, one more than that of an H_7. However, the node degree of a Q_7 is 7 and that of an H_7 is 6, while the diameter of an H_7 is 6, one less than that of a Q_7; that is, an H_7 can be favorably compared to a Q_7. It is interesting, however, to see that the diameter of a Q_{10} (a ten-dimensional hypercube) with 1024 nodes is 10, whereas that of an H_{19} with 1027 nodes is 18. This results from the fact that the node degree of hypercubes grows as $O(\log_2 p)$ while that of H-meshes remains constant (6), implying the existence of a tradeoff between the node degree and the communication diameter. Moreover, H-meshes have a finer scalability than hypercubes, i.e., $O(n^2)$ for H-meshes and $O(2^n)$ for hypercubes.

The complete binary tree structure offers the advantages of a constant node degree and an $O(\log_2 p)$ diameter. However, a tree is vulnerable to single link/node failures and suffers from the serious congestion of messages around its root [23]. These problems can be alleviated somewhat by adding additional links between leaf nodes. But the addition of links makes the optimal (shortest path) routing of messages very difficult and complicated [24]. Note that, although some other architectures such as chordal rings and square meshes [11], [12] also have a diameter of $O(p^{1/2})$ while their node degrees are less than that of an H_n, their lower connectivities will naturally lower the degree of fault tolerance. This is undesirable, especially when applications require the multiprocessor system to have a reasonably high degree of fault tolerance.

In general, an architecture strong in one aspect may be weak in the others; that is, no single architecture provides every desired feature. There are several types of architectures known to be suitable for interconnecting a large number of PN's, and H-meshes are one of them with many desirable features. Specifically, we showed that in view of their homogeneity, routing and broadcasting, fault tolerance, and implementability, H-meshes with the C-type wrapping are an attractive candidate architecture for interconnecting a large number of PN's.

VI. CONCLUSION

A systematic method for continuously wrapping H-meshes, called the C-type wrapping, is presented first. The C-type wrapping is then used to develop a simple addressing scheme, and efficient algorithms for routing and broadcasting for H-meshes. An H_3, called the HARTS (hexagonal architecture for real-time systems), is currently being built at the Real-Time Computing Laboratory, The University of Michigan. Each node consists of one to four M68020 processors configured as a shared memory multiprocessor. The nodes in HARTS are interconnected via custom-designed network processors, each of which is responsible for routing and buffering messages. The main objective of HARTS is to investigate various low-level architectural and fault-tolerant issues for critical real-time applications.

There are many interesting problems to be pursued for the H-mesh architecture, such as fault-tolerant routing in an H_n, embedding of interacting task modules into an H_n, and the application of the H-mesh to solve or reduce the complexity of some difficult problems. These topics are all closely related to the C-type wrapping and its associated routing and broadcasting algorithms and will be addressed in our future papers.

Acknowledgment

The authors would like to thank J. Dolter and P. Ramanathan at the RTCL for introducing us to the routing problem in H-meshes and their subsequent comments on an earlier draft of this paper. The authors are also indebted to A. van Tilborg, P. Hwang, and C. Holland at the Office of Naval Research for their encouragement and financial support.

References

[1] K. Hwang and J. Ghosh, "Hypernet: A communication-efficient architecture for constructing massively parallel computers," *IEEE Trans. Comput.*, vol. C-36, no. 12, pp. 1450-1466, Dec. 1987.
[2] W. D. Hillis, *The Connection Machine*. Cambridge, MA: MIT Press, 1985.
[3] T. Y. Feng, "A survey of interconnection networks," *IEEE Comput. Mag.*, pp. 12-27, Dec. 1981.
[4] C. Wu and T. Y. Feng, "On a class of multistage interconnection networks," *IEEE Trans. Comput.*, vol. C-29, no. 8, pp. 694-702, Aug. 1980.
[5] R. S. Wilkov, "Analysis and design of reliable computer networks," *IEEE Trans. Commun.*, vol. COM-20, no. 6, pp. 660-678, June 1972.
[6] G. A. Anderson and E. D. Jensen, "Computer interconnection networks: Taxonomy, characteristics and examples," *ACM Comput. Surveys*, vol. 7, pp. 197-213, Dec. 1975.
[7] W. A. Wulf, "HYDRA: The kernel of a multiprocessor operating system," *Commun. ACM*, vol. 17, June 1974.
[8] D. Mills, "An overview of the distributed computer network," in *Proc. Nat. Comput. Conf.*, vol. 45, 1976, pp. 523-531.
[9] K. S. Stevens, S. V. Robison, and A. L. Davis, "The Post Office—Communication support for distributed ensemble architectures," in *Proc. Six Int. Conf. Distrib. Comput. Syst.*, 1986, pp. 160-166.
[10] C. L. Seitz, "The Cosmic Cube," *Commun. ACM*, vol. 28, no. 1, pp. 22-33, Jan. 1985.
[11] T. Y. Feng, "Data manipulation functions in parallel processors and their implementations," *IEEE Trans. Comput.*, vol. C-23, no. 3, pp. 309-318, Mar. 1974.
[12] B. W. Arden and H. Lee, "Analysis of chordal ring network," *IEEE Trans. Comput.*, vol. C-30, no. 4, pp. 291-294, Apr. 1981.
[13] R. A. Finkel and M. H. Solomon, "Processor interconnection strategies," *IEEE Trans. Comput.*, vol. C-29, no. 5, pp. 360-370, May 1980.
[14] L. D. Wittie, "Communication structures for large networks of microcomputers," *IEEE Trans. Comput.*, vol. C-30, no. 4, pp. 264-273, Apr. 1981.
[15] A. J. Martin, "The Torus: An exercise in constructing a processing surface," in *Proc. 2nd Caltech Conf. VLSI*, 1981, pp. 527-537.
[16] M. Watkins, "Connectivity of transitive graphs," *J. Combinatorial Theory*, vol. 8, pp. 23-29, 1970.
[17] J. Turner, "Point-symmetric graphs with a prime number of points," *J. Combinatorial Theory*, vol. 3, pp. 136-145, 1967.
[18] S. L. Hakimi and I. T. Frisch, "An algorithm for construction of the least vulnerable communication network or the graph with the maximum connectivity," *IEEE Trans. Circuit Theory*, vol. CT-16, no. 5, pp. 229-230, May 1969.
[19] F. T. Boesch and R. E. Thomas, "On graphs of invulnerable communication nets," *IEEE Trans. Circuit Theory*, vol. CT-17, no. 5, pp. 183-191, May 1970.
[20] R. Halin, "A theorem on N-connected graphs," *J. Combinatorial Theory*, vol. 7, pp. 150-154, 1969.
[21] C. T. Ho and S. L. Johnsson, "Distributed routing algorithms for broadcasting and personalized communication in hypercubes," in *Proc. Int. Conf. Parallel Processing*, Aug. 1986, pp. 640-648.
[22] F. Harary, *Graph Theory*. Reading, MA: Addison-Wesley, 1969.
[23] E. Horowitz and A. Zorat, "The binary tree as an interconnection network: Applications to multiprocessor systems and VLSI," *IEEE Trans. Comput.*, vol. C-30, no. 4, pp. 247-253, Apr. 1981.
[24] J. R. Goodman and C. H. Sequin, "Hypertree: A multiprocessor interconnection topology," *IEEE Trans. Comput.*, vol. C-30, no. 12, pp. 923-933, Dec. 1981.

Ming-Syan Chen (S'87-M'88) was born in Taipei, Taiwan, Republic of China. He received the B.S. degree in electrical engineering from the National Taiwan University in 1982, and the M.S. and Ph.D. degrees in computer, information, and control engineering from The University of Michigan, Ann Arbor, in 1985 and 1988, respectively.

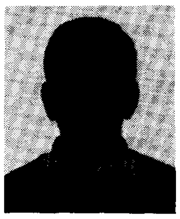
From 1982 to 1984, he served his military service in Taiwan as an electronics instructor. In Fall 1984, he went to The University of Michigan, Ann Arbor, where he was a research assistant in Real Time Computing Laboratory of EECS Department, and completed his graduate studies. Since fall 1988, he has been with IBM Thomas J. Watson Research Center, Yorktown Heights, NY, as a research staff member. His research interests include parallel computer architectures, especially hypercubes, distributed database systems, multiprocessor interconnection networks, algorithms, graph theory, and combinatorial theory.

Dr. Chen is a member of the Association for Computing Machinery.

Kang G. Shin (S'75-M'78-SM'83) received the B.S. degree in electronics engineering from Seoul National University, Seoul, 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 a Professor in the Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor, which he joined in 1982. He has been very active and authored/coauthored over 150 technical papers in the areas of fault-tolerant real-time computing, computer architecture, and robotics and automation. In 1985, he founded the Real-Time Computing Laboratory, where he and his students are currently building a 19-node hexagonal mesh multiprocessor, called HARTS, to validate various architectures and analytic results in the area of distributed real-time computing. From 1970 to 1972 he served in the Korean Army as an ROTC officer and from 1972 to 1974 he was on the research staff of the Korea Institute of Science and Technology, Seoul, working on the design of VHF/UHF communication systems. From 1978 to 1982 he was an Assistant Professor at Rensselaer Polytechnic Institute, Troy, NY. He was also a Visiting Scientist at the U.S. Airforce Flight Dynamics Laboratory in Summer 1979 and at Bell Laboratories, Holmdel, NJ, in Summer 1980. During the 1988-1989 academic year, he was a Visiting Professor in the Computer Science Division, University of California at Berkeley, and the International Science Institute, Berkeley, CA.

Dr. Shin was the Program Chairman of the 1986 IEEE Real-Time Systems Symposium (RTSS), the General Chairman of the 1987 RTSS and the Guest Editor of the 1987 August special issue of IEEE TRANSACTIONS ON COMPUTERS on Real-Time Systems. He is a distinguished speaker of the IEEE Computer Society. He is a member of Association for Computing Machinery, Sigma Xi, and Phi Kappa Phi. In 1987, he received the Outstanding Paper Award from the IEEE TRANSACTIONS ON AUTOMATIC CONTROL for a paper on robot trajectory planning.



Dilip D. Kandlur was born in Bombay, India, in 1961. He received the B.Tech. degree from the Indian Institute of Technology, Bombay, and the M.S.E. degree from the University of Michigan, Ann Arbor, in computer science and engineering.

He is currently a doctoral student and is working as a Research Assistant in the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor. His research interests include operating systems, parallel algorithms, and computer networks.