# Harvest Rate of Reconfigurable Pipelines

Weiping Shi, *Member, IEEE*, Ming-Feng Chang, and W. Kent Fuchs, *Fellow, IEEE*

**Abstract**—For a reconfigurable architecture, the harvest rate is the expected percentage of defect-free processors that can be connected into the desired topology. In this paper, we give an analytical estimation for the harvest rate of reconfigurable multipipelines based on the following model: There are $n$ pipelines each with $m$ stages, where each stage of a pipeline is defective with identical independent probability 0.5 and spare wires are provided for reconfiguration. By formulating the "shifting" reconfiguration as weighted chains in a partial ordered set, we prove when $n = \Theta(m)$, the harvest rate is between 34% and 72%.

**Index Terms**—Harvest rate, yield, reconfigurable arrays, defect tolerance, pipelines, random graphs, percolation.

## 1 Introduction

A multipipeline processor array is a set of one-dimensional pipelines running in parallel, where processors at different stages of the pipeline may be different; see Fig. 1. Multipipeline processor arrays are important for highly parallel architectures and vector supercomputer architectures [6]. VLSI and WSI technology makes it possible to fabricate a multipipeline processor array on a single chip or wafer. However, since it is likely that some processor elements will be defective, defect-tolerance for pipeline processor arrays can be important.

Harvest rate analysis for reconfigurable processor arrays is often difficult for two reasons. The first reason is that for most structures, it is NP-hard to compute the reliability; see Provan and Ball [9]. The second reason is that the reconfiguration algorithm may use complicated procedures to configure the system thereby making the resulting structure highly irregular. When all processors are of the same type, Greene and Gamal [2], Leighton and Leiserson [7], and other researchers have developed algorithms to reconfigure a single-pipeline array from a two-dimensional wafer. Their harvest rate analysis is only for extreme cases where the harvest rate either goes to 0 or goes to 1. Stornetta, Huberman, and Hogg [11] analyzed the harvest rate of multipipeline arrays, but since the problem is difficult, they used a phenomenological theory, combining analytical scaling equations with experimental measurements. Gupta, Zorat, and Ramakrishnan published an analysis of multipipeline processor arrays [5] based on the following technical assumption. They not only assumed each processor is defective with independent identical probability, but also assumed each processor is utilized with independent identical probability. They concluded that the harvest rate is independent of the shape of the array. To see the probability that one processor is utilized is related to the probability that an adjacent processor is utilized, consider the simplest example where the multipipeline array is a single pipeline with many stages. Therefore, one stage

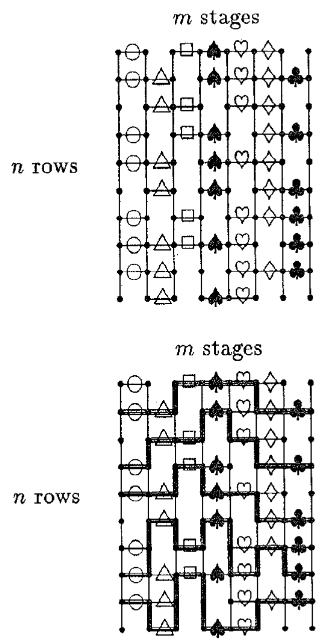


Fig. 1. A multipipeline array before and after reconfiguration.

can be utilized if and only if all other stages can be utilized. Other examples can be constructed accordingly. As a result, the probability that each stage is used is dependent on the probability that other stages can be used, no matter how one defines the probability space.

In this paper, we study the harvest rate of multipipeline processor arrays. There are $n \geq 1$ pipelines each with $m \geq 1$ stages. Each stage of a pipeline is defective with identical independent probability $p$. (This assumption, also used by [5] and [11], is a restriction of this model. However, we can take the maximum yield of all stages as $p$ to get an upper bound on the harvest rate of the array using the result of this paper. Similarly, we can take the minimum yield of all stages to get a lower bound on the harvest rate.) Vertical wires are provided for reconfiguration. We assume wires and switches are defect-free, an assumption also used by Greene and Gamal [2], Leighton and Leiserson [7], Gupta, Zorat, and Ramakrishnan [5], and many other researchers. The reconfiguration is done by routing around defective stages using vertical wires, and each vertical wire and switch can be used only once. Fig. 1 shows five horizontal pipelines before and after an example reconfiguration.

Our main focus is to analyze how many pipelines we can harvest on average if processors are defective at random. We will show the harvest rate $h(m, n)$, defined as the percentage of defect-free processors that can be connected into pipelines through the optimal reconfiguration, is between 34% and 72%, when $n$ is the same order as $m$. We formulate the "shifting" phenomenon of reconfiguration (some researchers call it fault stealing or compensation paths) as maximum weighted chains in a partial ordered set with random weights. Then, we use a mathematical result on the size of the chain to get our final result. Since the shifting phenomenon appears in many reconfiguration problems, we expect the method of analysis can be applied to other reconfigurable structures as well.

## 2 Analysis

To formulate the problem, define a rectangular graph $R(m + 1, n) = (V, E)$, where

- *W. Shi is with the Department of Computer Science, University of North Texas, Denton, TX 76203. E-mail: wshi@cs.unt.edu.*
- *M.-F. Chang is with the Department of Computer Science and Information Engineering, National Chiao-Tung University, Taiwan, Republic of China.*
- *W.K. Fuchs is with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907.*

*Manuscript received June 17, 1994; revised June 8, 1995.*
*For information on obtaining reprints of this article, please send e-mail to: transcom@computer.org, and reference IEEECS Log Number C95161.*

$$V = \{(i,j) : 1 \le i \le m+1 \text{ and } 1 \le j \le n\},$$

$$E = \{((i_1,j_1),(i_2,j_2)) : (i_1-i_2)^2 + (j_1-j_2)^2 = 1\}.$$

The set $\{(1, j) : 1 \le j \le n\}$ is called the left side, and the set $\{(m+1, j) : 1 \le j \le n\}$ is called the right side.

Define $R_{p,1}(m+1, n)$ as a random graph, where each horizontal edge appears with probability $p$, and each vertical edge appears with probability 1. Fig. 2 is an instance of $R_{p,1}(8, 10)$. It is clear that the number of pipelines we can harvest equals the maximum number of mutually vertex-disjoint paths from the left side to the right side in $R_{p,1}(m+1, n)$. In contrast to our Manhattan model, Gupta, Zorat, and Ramakrishnan [5] assumed a knock-knee model. However, the results for the two models are within a constant factor of 2.

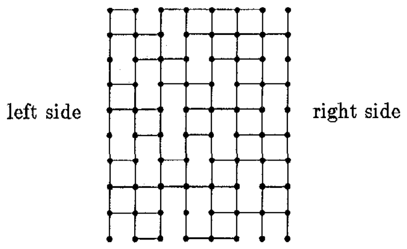


Fig. 2. An $R(8, 10)$ with some horizontal edges missing.

REMARK. The random graph $R_{p,p}(n, n)$ is known in percolation theory as the Bernoulli Square Lattice Bond Model [3]. However, since the vertical edge probability is 1, our problem is different from those studied in percolation theory.

The harvest rate $h(m, n)$ is defined as follows:

$$h(m,n) = \frac{E\begin{pmatrix}\text{number of vertex - disjoint}\\ \text{L - R paths in } R_{p,1}(m+1,n)\end{pmatrix}}{p \cdot n}.$$

The function $h(m, n)$ is well defined for all $m, n \ge 1$. It is easy to show $1 \ge h(m, n) \ge 0$ and $h(m, n)$ is monotonically decreasing in $m$, and monotonically increasing in $n$. We are interested in the asymptotic value of $h(m, n)$. The existence of the limit can be proved using a similar argument by Grimmett and Kesten [4]. Grimmett and Kesten proved that when both horizontal edges and vertical edges appear with probability $p$, then the limit exists [4]. However, estimating the limit is still an open problem in percolation theory.

For simplicity, we assume $p = 1/2$. The proof can be easily changed for any value of $p$. We first show that when the number of stages is too large compared to the number of pipelines, then the harvest rate is 0.

THEOREM 1. *For any constant $k$, if $m = \Omega(2^n/n^k)$, then $\lim_{m,n\to\infty} h(m, n) = 0$.*

PROOF. The probability that at least $\sqrt{n}+1$ edges in stage 1 are good is

$$\sum_{i=\sqrt{n}+1}^{n}\binom{n}{i}\frac{1}{2^n} = 1 - \sum_{i=0}^{\sqrt{n}}\binom{n}{i}\frac{1}{2^n} < 1 - \binom{n}{\sqrt{n}}\frac{1}{2^n}.$$

Therefore, the probability that each of the $m$ stages contains at least $\sqrt{n}+1$ edges is at most

$$\left(1-\binom{n}{\sqrt{n}}\frac{1}{2^n}\right)^m \le 2^{-\frac{m}{2^n}\binom{n}{\sqrt{n}}}.$$

Since

$$\frac{m}{2^n}\binom{n}{\sqrt{n}} = \frac{m}{2^n}\cdot\frac{n(n-1)\cdots(n-\sqrt{n}+1)}{\sqrt{n}(\sqrt{n}-1)\cdots 1}$$
$$> \frac{m}{2^n}\left(\frac{n}{\sqrt{n}}\right)^{\sqrt{n}} = \Omega\left(n^{\frac{\sqrt{n}}{2}-k}\right) \to \infty.$$

Therefore,

$$\lim_{m,n\to\infty} h(m,n) \le \frac{\sqrt{n}}{\frac{1}{2}n} = 0. \qquad \square$$

Now we present our main result. The reconfiguration shown in Fig. 1 is obtained by the greedy algorithm that always takes the bottom edge whenever possible, that is, if neither of the vertices on the left and right side of the edge was required in a previously built path. The following lemma proves the greedy reconfiguration always gives us the maximum number of pipelines after reconfiguration. Notice that we can obtain less interstage delay by distributing the pipeline stages evenly. However, since we are only concerned with the maximum number of pipelines, we use the greedy algorithm. See Libeskind-Hadas [8] for algorithms on reducing interstage delays.

LEMMA 1. *The greedy algorithm defined below can always find a maximum set of vertex-disjoint* L-R *paths.*

```
Algorithm.
Repeat
1) Take the lowest horizontal edge at each column;
2) Connect these edges into a path $P$;
3) Delete all horizontal edges in $P$ from the graph;
4) Delete all horizontal edges $((i, j), (i+1, j))$ from the
   graph if vertex $(i, j)$ or $(i+1, j)$ is in $P$.
Until step 1 fail.
```

PROOF. The proof is by induction on $k$, the number of vertex-disjoint paths in the graph. We also keep an invariant assertion that for every vertex $(i, j) \in P$, if there is no vertex $(i, j') \in P$ such that $j' > j$, then all vertical edges above $(i,j)$, i.e., edges

$((i, j), (i, j+1)), ((i, j+1), (i, j+2)), \ldots, ((i, n-1), (i, n)),$

have not been used so far.

When $k = 1$, the above algorithm can clearly find the path, and the invariant assertion holds. If there are $k$ vertex-disjoint paths in the graph, then shift the lowest path down to contain the bottom edge of each column. To do so we did not use any of the horizontal or vertical edges in the remaining $k-1$ paths. Since edges deleted in Steps 3 and 4 cannot be in any of the $k-1$ top paths, the remaining subgraph contains $k-1$ vertex-disjoint paths, and the invariant assertion holds. By the induction hypothesis, the algorithm will be able to find the remaining $k-1$ paths. $\square$

Lemma 1 allows us to study only the greedy reconfiguration algorithm. To describe the reconfiguration process, let random variable $y_{i,j}$ be the $Y$-coordinate of the $i$th horizontal edge in the $j$th L-R path; see Fig. 3.

Then it is easy to see

$$y_{ij} = \begin{cases} x_{i,j} & \text{if } j = 1\\ x_{i,j} + \max\{y_{i,j-1}, y_{i+1,j-1}\} & \text{if } j > 1, i = 1\\ x_{i,j} + \max\{y_{i,j-1}, y_{i-1,j-1}, y_{i+1,j-1}\} & \text{if } j > 1, m > i > 1\\ x_{i,j} + \max\{y_{i,j-1}, y_{i-1,j-1}\} & \text{if } j > 1, i = m\end{cases}$$

where $x_{i,j}$s are independent random variables having the same geometric distribution:

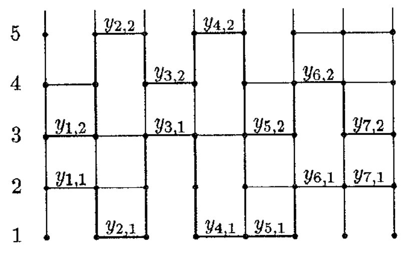


Fig. 3. Random variables of two lowest paths.

$$\Pr\{x_{i,j} = k\} = \left(\frac{1}{2}\right)^k, \quad \text{for } k = 1, 2, \ldots$$

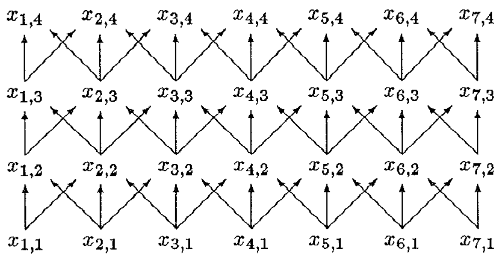
Intuitively, $x_{ij}$ is the distance we have to move up to find the $i$th stage of the $j$th path, from the $(j-1)$th path. Let $Y_k = \max_{i=1}^{m}\{y_{i,k}\}$, then there are $k$ vertex-disjoint L-R paths if and only if $Y_k \le n$. Therefore the problem of estimating the number of pipelines becomes the problem of estimating the random variable $Y_k$.

To estimate $Y_k$ directly is hard, because $y_{i,j}$s are not mutually independent, and are defined recursively. However, since $x_{i,j}$s are mutually independent, we construct a directed graph $D(k) = (X, E)$, where $X = \{x_{i,j}$: $1 \le i \le m$ and $1 \le j \le k\}$ and $E$ is a set of directed edges defined as follows. For every vertex $x_{i,j}$, there are edges from $x_{i,j}$ to

$$\begin{cases} \varnothing & \text{if } j = k \\ x_{i,j+1}, x_{i+1,j+1} & \text{if } j < k \text{ and } i = 1 \\ x_{i-1,j+1}, x_{i,j+1}, x_{i+1,j+1} & \text{if } j < k \text{ and } 1 < i < m \\ x_{i-1,j+1}, x_{i,j+1} & \text{if } j < k \text{ and } i = m. \end{cases}$$

See Fig. 4. Now, $y_{i,j}$ becomes the value of the maximum-weighted directed path to $x_{i,j}$ in the graph $D(k)$, where each vertex $x_{i,j}$ has a geometrically distributed weight $w(x_{i,j})$, and

$$Y_k = \max_{1\le i\le m}\{y_{i,k}\} = \left(\begin{matrix}\text{maximum weighted}\\ \text{directed path in } D(k)\end{matrix}\right).$$



$x_{1,4}$ $x_{2,4}$ $x_{3,4}$ $x_{4,4}$ $x_{5,4}$ $x_{6,4}$ $x_{7,4}$
$x_{1,3}$ $x_{2,3}$ $x_{3,3}$ $x_{4,3}$ $x_{5,3}$ $x_{6,3}$ $x_{7,3}$
$x_{1,2}$ $x_{2,2}$ $x_{3,2}$ $x_{4,2}$ $x_{5,2}$ $x_{6,2}$ $x_{7,2}$
$x_{1,1}$ $x_{2,1}$ $x_{3,1}$ $x_{4,1}$ $x_{5,1}$ $x_{6,1}$ $x_{7,1}$

Fig. 4. Directed graph $D(k)$ with $k = 4$.

To estimate the maximum-weighted path in $D(k)$, it first seems that if we add the maximum $x_{i,j}$ from each row, it might give us an upper bound. Unfortunately, this bound will not be good enough. The key here is to use the underlying combinatorial structure to argue that the maximum weighted directed paths cannot be too large.

Embed $D(k)$ into grid $L(m + k - 1, m + k - 1)$, see Fig. 5, and assign each element $x$ in $L$ with an integer random variable $w(x)$ that has a geometric distribution with parameter 1/2. The grid $L(m + k - 1, m + k - 1) = \{(i, j) \mid 1 \le i, j \le m + k - 1\}$ is a special kind of *partially ordered set*. A partially ordered set, or poset, is a set of elements and a binary relation $>$ that is reflexive, antisymmetric, and transitive. In $L$, two elements, $a = (x_a, y_a)$ and $b = (x_b, y_b)$, have the binary relation $a > b$ if $x_a \ge x_b$ and $y_a \ge y_b$. A *chain* in a poset is a set of pairwise comparable elements. In $L$, a chain is a set of elements

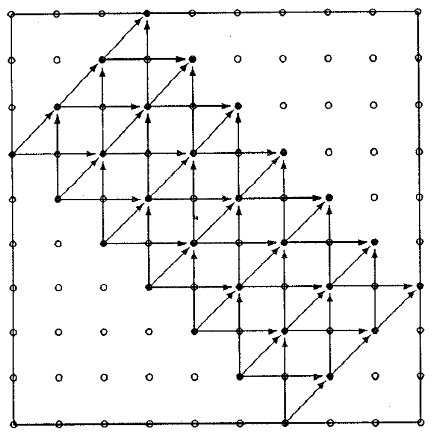


Fig. 5. Embed $D(k)$ on grid $L(m + k - 1, m + k - 1)$.

that form a monotonically increasing path. A maximum chain is a chain that contains the maximum number of elements, and a maximum weighted chain is a chain such that the sum of weights of its elements is maximum. See Aigner [1] for more on the basic concepts of combinatorics. For our problem, we want to find the maximum weighted chain in the poset $D$.

LEMMA 2. *If $k = \Theta(m)$, then the maximum weighted chain in $D$ is of expected size $k/c$, where $0.36 \ge c \ge 0.17$. Furthermore, the probability that the maximum weight chain in $D$ is greater than $k/0.17$ or less than $k/0.36$ is $o(1/n)$.*

Due to the page limitation, we omit the proof, which can be found in a technical report [10].

THEOREM 2. *If $n = \Theta(m)$, then $\lim_{m,n\to\infty} h(m, n)$ is between 0.34 and 0.72.*

PROOF. Since the height of the $k$th path is $Y_k$, when $Y_k < k/0.17 = n$, there will be $k$ paths, and this happens with high probability. Therefore the harvest rate is at least

$$k/\left(\tfrac{1}{2}n\right) = k/\left(\tfrac{1}{2}k/0.17\right) = 0.34.$$

On the other hand, when $Y_k > k/0.36 = n$, there will not be $k$ paths and this happens with high probability. Therefore, the harvest rate is at most 0.72.

Simulation in Table 1 shows how the harvest rate changes as the wafer size increases. Each row of the table is computed by averaging 1,000 random grid graphs. Notice when $m$ and $n$ increases, $h(m, n)$ also increase. This is because on the boundary, defective processors are not as easily repairable as in the center, and as the array size increases, the boundary portion decreases.

TABLE 1
SIMULATION RESULTS

| # of Stages $m$ | # of Pipelines $n$ | Harvest Rate $h(m, n)$ |
|---|---|---|
| 10 | 10 | 0.483 |
| 20 | 20 | 0.484 |
| 30 | 30 | 0.513 |
| 40 | 40 | 0.499 |
| 50 | 50 | 0.511 |
| 60 | 60 | 0.520 |
| 70 | 70 | 0.524 |
| 80 | 80 | 0.531 |
| 90 | 90 | 0.532 |

## 3 Conclusion

In this paper, we analyzed the harvest rate of reconfigurable multipipeline processor arrays. We showed that the "shifting" or "fault stealing" phenomenon during reconfiguration can be described as the maximum weighted chains in a poset with random weights, and we used a combinatorial argument to give a bound on the size of the maximum weighted chain. Our method is the first purely analytical approach to analyzing reconfiguration of linear arrays. We propose as an open problem to find the exact value of $h(m, n)$.

## Acknowledgments

The authors thank Douglas B. West for valuable discussions concerning this research, Ran Libeskind-Hadas for suggestions, Harry Kesten, Peter Winkler for providing references, and an anonymous referee (D) for improving the presentation. This research was supported in part by the SDIO/IST and managed by the U.S. Office of Naval Research under contract N00014-89-K-0070. Weiping Shi was also supported in part by the U.S. National Science Foundation under grant MIP-9309120.

## References

[1] M. Aigner, *Combinatorial Theory*. Springer-Verlag, 1979.
[2] J.W. Greene and A. Gamal, "Configuration of VLSI Arrays in the Presence of Defects," *J. ACM*, vol. 41, no. 4, pp. 694–717, 1984.
[3] G. Grimmett, *Percolation*. New York: Springer-Verlag, 1989.
[4] G. Grimmett and H. Kesten, "First Passage Percolation, Network Flows and Electrical Resistances," *Z. Wahrscheinlichkeitstheor.* Verw 66, pp. 335–366, 1984.
[5] R. Gupta, A. Zorat, and I. V. Ramakrishnan, "Reconfigurable Multipipelines for Vector Supercomputers," *IEEE Trans. Computers*, vol. 38, no. 9, pp. 1,297–1,307, Sept. 1989.
[6] K. Hwang, *Advanced Computer Architecture: Parallelism, Scalability, Programmability*. New York: McGraw-Hill, 1993.
[7] F.T. Leighton and C.E. Leiserson, "Wafer-Scale Integration of Systolic Arrays," *IEEE Trans. Computers*, vol. 34, no. 5, pp. 448-461, May 1985.
[8] R. Libeskind-Hadas, "Reconfiguration of Fault Tolerant VLSI Systems," PhD thesis, Dept. of Computer Science, Univ. of Illinois at Urbana-Champaign, Oct. 1993.
[9] J.S. Provan and M.O. Ball, "The Complexity of Counting Cuts and of Computing the Reliability That a Graph Is Connected," *SIAM J. Computing*, vol. 12, no. 4, pp. 777–788, Nov. 1983.
[10] W. Shi, "Design, Analysis and Reconfiguration of Defect-Tolerant VLSI and Parallel Processing Arrays," PhD thesis, Dept. of Computer Science, Univ. of Illinois at Urbana-Champaign, June 1992. Coordinated Science Laboratory Technical Report CRHC-94-21, Sept. 1994.
[11] W.S. Stornetta, B.A. Huberman, and T. Hogg, "Scaling Theory for Fault Stealing Algorithms in Large Systolic Arrays," *IEEE Trans. Computer-Aided Design*, vol. 9, no. 3, pp. 290–298, Mar. 1990.

# Load Sharing in Hypercube-Connected Multicomputers in the Presence of Node Failures

Yi-Chieh Chang and Kang G. Shin, *Fellow, IEEE*

**Abstract**—This paper addresses two important issues associated with load sharing (LS) in hypercube-connected multicomputers: 1) ordering fault-free nodes as preferred receivers of "overflow" tasks for each overloaded node and 2) developing an LS mechanism to handle node failures. Nodes are arranged into *preferred lists* of receivers of overflow tasks in such a way that each node will be selected as the *k*th preferred node of one and only one other node [1]. Such lists are proven to allow the overflow tasks to be evenly distributed throughout the entire system. However, the occurrence of node failures will destroy the original structure of a preferred list if the failed nodes are simply dropped from the list, thus forcing some nodes to be selected as the *k*th preferred node of more than one other node. We propose three algorithms to modify the preferred list such that its original features can be retained regardless of the number of faulty nodes in the system. It is shown that the number of adjustments or the communication overhead of these algorithms is minimal. Using the modified preferred lists, we also proposed a simple mechanism to tolerate node failures. Each node is equipped with a backup queue which stores and updates the information on the tasks arriving/completing at its most preferred node.

**Index Terms**—Load sharing, hypercube-connected multicomputers, real-time systems, node failures, backup queues.

## 1 Introduction

Load sharing (LS) in general-purpose distributed systems has been studied extensively by numerous researchers and many LS algorithms proposed [2], [3], [4], [5]. These LS algorithms are usually designed to minimize the average task-response time. By contrast, LS in distributed real-time systems has been addressed far less than that in general-purpose distributed systems.

In [6], we have proposed a decentralized, dynamic LS method for real-time applications. In this method, each node maintains the state of a set of nodes in its proximity, called a *buddy set*. Three thresholds of queue length (QL), denoted by $TH_u$, $TH_f$, and $TH_v$, are used to define the (load) state of a node. A node is said to be *underloaded* if $QL \leq TH_u$, *medium-loaded* if $TH_u < QL \leq TH_f$, *fully-loaded* if $TH_f < QL \leq TH_v$, and *overloaded* if $QL > TH_v$. Whenever a node becomes fully-loaded due to the arrival and/or transfer of tasks, it will broadcast this change of state to all the nodes in its buddy set; so will it when a node becomes underloaded as a result of completing the execution of tasks. Every node that receives this state-change broadcast will update its state information by marking the node as fully-loaded or underloaded in its ordered list (called a *preferred list*) of available receivers. When a node becomes overloaded, it can then select, without probing other nodes, the first underloaded node from its preferred list. Note that the preferred list of each node does not change over the time, but the nodes will be dynamically marked as underloaded or overloaded according to their load states, so that an overloaded node may select the first underloaded node from its preferred list.

• *The authors are with the Real-Time Computing Laboratory, Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109-2122. E-mail: kgshin@eecs.umich.edu.*

*Manuscript received July 11, 1994; revised July 10, 1995.*
*For information on obtaining reprints of this article, please send e-mail to: transcom@computer.org, and reference IEEECS Log Number C95165.*