

Randomized Routing and PRAM Emulation on Parallel Machines

David S.L. Wei

Department of Computer and Information Science
Fordham University
Bronx, NY 10458-5198
E-mail: wei@cs.fordham.edu

Abstract. This paper shows the power of randomization in designing efficient parallel algorithms for the problems of routing and PRAM emulation. We show that with randomization techniques optimal routing can be obtained for a large class of processor interconnection networks called *leveled networks*. This class includes well-known networks (e.g., mesh, hypercube and de Bruijn) as well as new ones (e.g., star graph, for which the network diameter is *sub-logarithmic* in the network size). Using the obtained routing algorithms, optimal emulations of the CRCW PRAM on the leveled network and mesh are also given.

1 Introduction

Since the time [14] and [16] introduced the technique of randomizing an algorithm to improve its efficiency, randomization has been successfully used to solve numerous computational problems. There are two types of randomized algorithms: 1) those that always output the correct answer but whose running time is a random variable with a specified mean. These are called Las Vegas algorithms; and 2) those that run for a specified amount of time and whose output will be correct with a specified probability. These are called Monte Carlo algorithms. Hoare's quicksort algorithm [4] is of the first type, and Primality testing algorithm of Rabin [14] is of the second type.

Advantages of randomized algorithms are mainly twofold, namely simplicity and efficiency. The majority of the randomized algorithms found in the literature are much simpler and easier to understand than the best deterministic algorithms for the same problems. Randomized algorithms have also been shown to yield better complexity bounds. In this paper we survey some randomized routing and PRAM emulation algorithms, as both problems are vital in the area of parallel processing.

2 Preliminaries

2.1 Models Definition

Though randomized parallel algorithms can be developed on a variety networks, we will employ only the following networks for the discussion.

Star Graph Let $s_1 s_2 \dots s_n$ be a permutation of n symbols, *e.g.*, $1 \dots n$. For $1 < j \leq n$, we define $SWAP_j(s_1 s_2 \dots s_n) = s_j s_2 \dots s_{j-1} s_1 s_{j+1} \dots s_n$. An n -star graph is a graph $S_n = (V, E)$ with $|V| = n!$ nodes, where $V = \{s_1 s_2 \dots s_n \mid s_1 s_2 \dots s_n \text{ is a permutation of } n \text{ different symbols}\}$, and $E = \{(u, v) \mid u, v \in V \text{ and } v = SWAP_j(u) \text{ for some } j, 1 < j \leq n\}$. It is not hard to see that the degree of an n -star graph is $n - 1$. Also, in [1], Akers, Harel, and Krishnamurthy have shown that the diameter of an n -star graph is $\lfloor \frac{3}{2}(n-1) \rfloor$ which is sublogarithmic in the size of the network.

Mesh A two dimensional mesh has $N = n \times n$ nodes interconnected in the form of an $n \times n$ grid and has a degree of 4. Clearly, its diameter is $2\sqrt{N} - 2$.

Leveled Networks An (N, ℓ) *leveled network* consists of $\ell + 1$ groups of nodes such that each group has N nodes and these groups form a sequence of $\ell + 1$ columns (one group per column), say $c_1, c_2, \dots, c_{\ell+1}$. Column c_1 and column $c_{\ell+1}$ are identified; thus, although there are $\ell + 1$ columns of N nodes each, the total number of nodes is ℓN . The only links in the network are between nodes in c_i and nodes in either c_{i+1} or c_{i-1} (provided these columns exist). Every node in each column has at most d incoming and outgoing links, where d is the degree of the network. For each node in the first column, there exists a unique path of length ℓ connecting it to any node in the last column. Clearly, the diameter of a leveled network is ℓ .

A leveled network is called **nonrepeating** if it satisfies the following property: if any two distinct paths from the first column to the last column share some links and then diverge, these two paths will never share a link again. Leveled networks are interesting because the problem of packet routing in various single-stage ICNs (such as the n -cube) can be reduced to an equivalent packet routing problem on a leveled network. The n -cube, de Bruijn network, star graph, mesh, and a host of other single-stage networks can all be represented as nonrepeating leveled network.

PRAM A parallel random-access machine (PRAM) [5] is an abstract parallel computer model consisting of an arbitrary number of processors that communicate via a shared global memory. Each memory access to the shared memory is assumed to take unit time. This *unit-time memory access* property simplifies programming because it permits parallel algorithms to be designed and analyzed solely on the basis of their *computational* requirements, divorced from issues of interprocessor communication. EREW PRAM is the model where no simultaneous read or write is allowed on any cell of the shared memory. And CRCW PRAM model allows both concurrent read and concurrent write.

2.2 Some Facts from Probability Theory

Chernoff bound Let X stand for the number of heads in n independent flips of a coin, the probability of a head in a single flip being p . X is also known to

have a binomial distribution $B(n, p)$. The following fact is known as Chernoff bound:

$$\Pr[X \geq m] \leq \left(\frac{np}{m}\right)^m e^{m-np}, m > np.$$

3 Randomized Routing

The routing problem is defined as follows. We are given a specific network and a set of packets of information where a packet is a $\langle source, destination \rangle$ pair. To start with, the packets are placed on their sources. These packets must be routed in parallel to their own destinations such that at most one packet passes through any link of the network at any time and all packets arrive at their destinations as quickly as possible. A paradigmatic case of general routing is **permutation routing** in which initially there is exactly one packet at each node and the destinations form some permutation of the sources. A routing algorithm is said to be **oblivious** if the path taken by each packet depends only on its own source and destination. A routing algorithm is **non-oblivious** otherwise. An oblivious routing strategy is preferred, since it leads to a simple control structure for the individual processing elements. If an oblivious routing algorithm is **deterministic**, then it means that for each

$\langle source, destination \rangle$ pair, there is a *unique* path which any packet with that $\langle source, destination \rangle$ specification must take. Borodin and Hopcroft [3] have shown that if we insist on both obliviousness and determinism in a permutation routing, then we have to pay a heavy price for it, namely a running time of $\Omega(\sqrt{\frac{N}{d}})$ for a network of N nodes with degree d . This fact suggests that for networks with small diameters, if one wants to route any permutation request within a small constant factor of the diameter of the network, one has to give up either obliviousness or determinism or both. Fortunately, randomization has been successfully used in the design of optimal oblivious routing algorithms¹ for various networks following the two-phase routing scheme of Valiant [19, 18]. We use \tilde{O} to represent the complexity bounds of randomized algorithms. We say a randomized algorithm has resource (like time, space etc.) bound of $\tilde{O}(g(n))$ if there exists a constant c such that the amount of resource used by the algorithm (on any input of size n) is no more than $cg(n)$ with probability $\geq 1 - \frac{1}{n^c}$.

We first define the **path** (route) of a packet x as the sequence of nodes and links that the packet x ever travels. Also, we define the **delay** of a packet x in a run of a routing scheme as the total number of time units during which x waits unserved in queues of nodes along its path. The performance of any routing scheme is usually assessed in terms of its **routing time**, **queue size**, and **queueing discipline**. The routing time is the number of steps taken by the last packet to reach its destination. The number of steps taken by a packet x is simply the sum of the *delay* of x and the length of the *path* of x . It determines

¹ An optimal randomized routing algorithm is an algorithm that runs within a small constant factor of the diameter of the network with high probability.

how fast routing can be finished. The maximum number of packets residing at any node at any time step during the entire course of routing determines the queue size of a routing scheme, and consequently determines the amount of additional hardware needed per node. The queueing discipline is a strategy of the processors in the network to assign priority to the packets queued. The first-in first-out (FIFO) is a simple queueing strategy and, thus, is preferred.

Queue Line vs Delay Path

Queue line lemma

A **queue line** associated with a packet x is a directed *path* taken by x , together with the packets that *overlap* with x . (Two packets are said to overlap if there are ≥ 1 common links in their paths.)

The number of steps a packet x is delayed is less than or equal to the number of packets that *overlap* with x provided the routing scheme is nonrepeating.

Delay path

Suppose the packet y_m is one of the latest finished packets with destination x_m in a run of a routing scheme. Let y_{m-i} be the packet which delays the packet $y_{m-(i-1)}$ at node x_{m-i} , $\forall i, 1 \leq i < m$ (A packet y_i is delayed by packet y_j at node x either because both y_i and y_j are in node x and y_j has a higher priority or because y_i is in x and the queue that y_i wants to go next is full and y_j is in the head of that queue). Then the collection of nodes x_1, \dots, x_m , and the nodes along the path of y_i between pair of nodes (x_i, x_{i+1}) , $\forall i, 1 \leq i < m$, together with links along the path formed by these nodes forms the **delay path** of the sequence of packets y_1, \dots, y_m .

Most literature use either **queue line** or **delay path** in analyzing the performance of their routing schemes. If an analysis for a routing scheme is based on queue line lemma, the way to prove an upper bound on the routing time, say, $O(\delta)$, can be briefly sketched as follows. Since the path taken by a packet during the entire course of routing is, in fact, a queue line, those N paths taken by all N packets form a set of non-disjoint queue lines. One can try to prove that according to the randomization embedded in the algorithm, the probability that there exists at least one queue line which *overlaps* with more than δ other queue lines is bounded above by a very small value, say, $\frac{1}{N^\epsilon}$. The advantage of using the queue line lemma in the proof is that it leads to a simpler analysis. However, in the analysis of routing schemes with constant queue size, it seems that queue line lemma is not strong enough. Although the idea of a delay path is not as simple as that of a queue line, it may lead to a successful proof of a better performance. After Upfal introduced the idea of delay path [17], Ranade [15], and Leighton, Maggs and Rao [9] used a notion similar to delay path to prove that their routing schemes need only a queue of size $O(1)$. We provide a brief sketch of their proof. Let \mathfrak{R} be the total number of delay paths in a run of a routing scheme, and \mathbf{D} be the possible number of delay paths in which the last packet takes $\geq \delta$ steps to finish. Then the probability that routing takes $\geq \delta$ steps to finish is simply the ratio of \mathbf{D} to \mathfrak{R} , i.e. $Prob(T \geq \delta) = \frac{\mathbf{D}}{\mathfrak{R}}$.

There has been great success in the development of efficient randomized routing schemes. By employing randomization in a routing algorithm, one can

achieve oblivious permutation routing within a small constant factor of the diameter of the network. The research work in this direction is pioneered by Valiant and Brebner [19]. The main contribution of their work is shown in the following theorem.

Valiant and Brebner's Theorem [19] *Given a network with (1) N nodes, (2) degree d , and (3) diameter μ , any permutation routing on the network can be performed by an oblivious, nonrepeating, and symmetric randomized routing algorithm in δ steps with probability $\geq 1 - \left(\frac{e\mu^2}{\delta d}\right)^\delta N$, where $e = 2.71 \dots$.*

To prove this theorem, they first fixed a path of length r taken by a given packet, say y . Then based on the assumption of symmetry, they proved that the summation of the probability of each of the N (including one dummy trial) packets to share at least one link with the fixed path is $\leq \frac{r\mu}{d}$. Using **Hoeffding** bound, the probability of having at least δ successful packets is $\leq B(\delta, N, \frac{r\mu}{Nd})$ provided $\delta \geq \frac{r\mu}{d} + 1$. Then based on Chernoff bound and queue line lemma, the probability that y suffering a delay of at least δ is bounded by $\left(\frac{e\mu^2}{\delta d}\right)^\delta$. Since there are N packets to be routed, the probability that at least one of them suffers a delay of at least δ is $\leq \left(\frac{e\mu^2}{\delta d}\right)^\delta N$.

By making use of this theorem, they gave an $\tilde{O}(\log N)$ time oblivious randomized routing scheme for the n -cube network of $N = 2^n$ nodes. One deficiency of their analysis is that the analysis won't work for any constant degree network. However, they conjectured that there exist optimal randomized routing algorithms for some constant degree networks. This conjecture was proven by Upfal [17], and independently by Aleliunas [2]. Upfal developed the notion of *delay path* and showed the way to route on the butterfly in $\tilde{O}(\log N)$ steps using queues of size $O(\log N)$. Aleliunas also used a similar idea to obtain an equivalent result on d -way shuffle (de Bruijn network). Both of their algorithms use *priority queues*. The idea behind the proof of Upfal's algorithm is to bound the probability of the event that there exists at least one delay path whose last packet is finished in $\geq c \log N$ steps. Another deficiency of Valiant and Brebner's analysis is that if a given network has *sublogarithmic* diameter, the theorem is not able to guarantee an optimal routing on it. Palis et al. settle this problem in [11]. They show that for the n -star graph with $N = n!$ nodes, any permutation routing can be completed by a randomized routing algorithm in $\tilde{O}(n) = \tilde{O}(\log N / \log \log N)$ steps. A summary of their algorithm and analysis follows. The routing algorithm is an adoption of Valiant's two-phase randomized routing. During phase 1, each packet selects a random intermediate node and is sent by a greedy routing algorithm (described in what follows). And in the second phase, each packet is forwarded from the random node to its true destination along the unique path determined by the greedy routing scheme. Let G^i be a subgraph with the last i symbols of the labels of all nodes in it be identical (G^i is itself an $(n - i)$ -star graph, $0 \leq i < n$). The greedy routing can be viewed as a sequence of stage transitions $S_0 S_1 \dots S_i S_{i+1} \dots S_{n-1}$, where in S_i the packet is in G^i to which the destination of the packet belongs. Routing then turns out to be forwarding the

packets along the leveled network representation of the star graph in the sense that the links from column (level) $i - 1$ to column i are the links (of the star graph) that can be used during the transition from S_{i-1} to S_i . In the leveled network representation of the star graph, each node in column i has $n - i + 1$ incoming and $n - i$ outgoing links. Let π be an arbitrary packet in the network. The upper bound on the delay π suffers is computed as follows. Let d_i stand for the number of packets that meet π 's path for the first time at level i (for $1 \leq i \leq \ell$). Since the leveled network representation of the star graph is non-repeating, the queue line lemma can be used to infer that an upper bound for the delay π suffers is $\sum_i d_i$. This can be computed using generating functions.

The generating function for $Prob(d_i = d)$ is calculated as $G_i(x) = e^{\frac{x}{n^2}}$. The generating function (call it $G(x)$) for $Prob(\sum_i d_i = d)$ is the product of the G_i 's. $G(x)$ simplifies to $e^{\frac{4n}{n^2}x}$. This immediately implies that the probability that the total delay is greater than a given amount, say δ , is:

$$Prob(\sum_i d_i \geq \delta) \leq \sum_{d=\delta}^{\infty} \left(\frac{4n}{n^2}\right)^d \frac{1}{d!}.$$

If δ is chosen to be equal to can for some appropriate constant c , the above probability can be shown to be no more than $N^{-\alpha}$. As a corollary, they also show that the same time complexity also holds for a d -ary de Bruijn network.

Pippenger [13] improved the works of [17] and [2] and obtained a constant queue size optimal routing algorithm on a butterfly. However, his algorithm as well as the accompanying analysis are quite complicated. The algorithm also allows a small possibility of deadlock. Ranade [15] used a new idea to obtain a simpler optimal algorithm, and used a new proof technique to have a much simpler analysis for the algorithm. He introduced the notion of a **polarized sequence**, which is similar to the idea of a *delay path*, and showed that a large delay ($\geq c \log N$) will occur only when there is a long sequence of reverse order packets polarized along a short path². Then he showed that when his algorithm runs on the butterfly, the probability that a long polarized sequence occurs is quite low. Since his algorithm requires the routed packets to be in sorted order along the same direction of the traveled path, a combining technique can be employed by the algorithm to handle the *many-one* routing problem. Consequently, together with an address mapping technique (hashing), his algorithm can also be used to simulate a step of an N node CRCW PRAM on an N node butterfly in $\tilde{O}(\log N)$ steps using FIFO queues of constant size.

As a consequence of Ranade's work, research in the direction of finding an optimal routing algorithm for a specific constant degree cube class network has reached a dead end. However, the proposed routing algorithms as well as their accompanying analysis are network-specific. The research on routing problems is thus switched to find a network-independent algorithm for a large class of networks. The research in this new direction was pioneered by Leighton, Maggs,

² A short path here means a path with length $\leq 8n$, where n is the diameter of a butterfly network with $N = n2^n$ nodes.

and Rao [9]. They generalized Ranade's algorithm for a class of *leveled networks*. For any leveled network with constant degree, ℓ levels, and congestion c (the expected maximum number of distinct packets that will pass through a link during the entire course of routing), their algorithm can perform any permutation of N packets in time $\tilde{O}(\ell + c + \log N)$. The analysis required for the proof of the algorithm is simpler than that of Ranade's, and the queue size is quite small (even could be as small as 1) which is impossible in Ranade's analysis. Their algorithm can also be used in the $n \times n$ mesh and runs in $\tilde{O}(n)$ steps with constant queue size. Randomized routing on mesh is also addressed by Krizanc, Rajasekaran, and Tsantilas [7]. This work predates those of [8] and [10]. They gave a randomized oblivious algorithm for the $n \times n$ mesh that runs in $2n + \tilde{O}(\log n)$ steps using queues of size $O(\log n)$. They also modified this algorithm to obtain a nonoblivious algorithm with the same running time using queues of constant size. An open question is thus introduced here: Can one perform any permutation routing in *sublogarithmic* steps for non-constant degree leveled networks with $\ell, c \leq \log N$? This question was answered by Palis et al. in [12]. They show that for an (N, ℓ) nonrepeating leveled network, any permutation routing of N packets (from the first column to the last column) can be completed in $\tilde{O}(\ell)$ steps provided that $d \geq 2$ (where d is the degree of the network) and $\ell = \Omega(\log N / \log \log N)$. For each link, the queue size is $O(\ell)$ with high probability. A summary of their algorithm and analysis follows. The routing algorithm is also an adoption of Valiant's two-phase randomized routing. During phase 1, each packet from the first column of the leveled network is sent to a random node in the last column by traversing a random link at each level. During phase 2, each packet is sent from the random node to its true destination along the unique path. The queuing discipline is FIFO. The analysis is also using the generating functions and is similar to that of the star graph routing.

4 Randomized PRAM Emulation

4.1 PRAM Emulation on any Interconnection Network (ICN)

In [12], Palis et al. consider the problem of emulating an EREW PRAM with N processors and shared memory of size M on an N -node ICN. The emulation result can be extended to a CRCW PRAM using 'message combining' (see [15] [20]). Their emulation algorithm is based on Karlin and Upfal's technique called *parallel hashing* [6]. The idea is to map the M shared memory cells of the PRAM onto the local memory modules of the N processors of the ICN. The mapping is obtained by randomly choosing a hash function h from the following class of hash functions:

$$H = \{h | h(x) = ((\sum_{0 \leq i < \rho} a_i x^i) \bmod P) \bmod N\}$$

where P is a prime, $P \geq M$, $a_i \in Z_P$, and ρ depends on N .

The above class of hash functions has the following interesting property:

Fact [6] If N items are mapped into $N/2^i$ buckets using a random hash function (from the class defined before), the maximum number (call it Y_i) of items mapped into a single bucket satisfies:

$$\text{Prob.}[Y_i \geq j] \leq N \left(\frac{2^i}{j - \rho} \right)^\rho.$$

Consider the case of distributing N packets among N processors using the above hashing scheme. The above fact implies that if ρ is chosen to be some constant multiple of $\log N / (\log \log N)$, then each processor will get $\tilde{O}(\rho)$ packets.

Given the above address mapping, the memory requests (read or write) of the PRAM processors can be simulated on the ICN as follows. Suppose that PRAM processor i wants to access shared memory location j . On the ICN, this step is accomplished in two phases: (1) processor i sends a packet (encoding the request) to processor $h(j)$; and (2) if the packet was a ‘read’ request, processor $h(j)$ sends the contents of memory cell j back to processor i . Each of these two phases corresponds to a routing task. In the first phase, there is at the most one packet originating from any node and there are at most $O(\log N / (\log \log N))$ packets destined for any node w.h.p, whereas in the second phase there are at the most $O(\log N / (\log \log N))$ packets starting from any node and at most one packet destined for any node.

4.2 PRAM emulation on a leveled network

Palis et al. [12] make use of the above address mapping for PRAM emulation on the leveled networks. A single step of the PRAM is simulated as follows. Let processor i want to access memory cell j . On the leveled network: 1) processor i in the first column sends a packet (encoding the request) to processor $h(j)$ in the last column (which, recall, coincides with the first column); 2) if the request was a ‘read’, $h(j)$ sends the contents of cell j back to processor i . Without loss of generality it is assumed that nodes in the first column are processors and nodes in the last column are memory modules numbered as $0, \dots, N - 1$. The routing algorithm used for communication is the one introduced in the previous section. Suppose S is the set of items being requested by the processors for executing a PRAM instruction, $|S| \leq N$. If it can be proved that with very high probability (say $1 - \frac{1}{N^c}$, c' being a constant > 0), no more than $O(1)$ items from S will be mapped onto the same memory module, then the routing algorithm in the previous section together with its analysis can be directly used to prove the desired performance of the emulation. Unfortunately, with probability $N^{-\beta}$ (for some $\beta > 0$), at least one node will get $c\ell$ (for some constant c) items. However, even if there are $c\ell$ items to be mapped into each memory module, the desired performance can still be obtained. In order to obtain the desired performance, same routing algorithm was used, but the analysis was different. They first prove that their algorithm for the leveled networks can perform a *partial ℓ -relation* routing in $\tilde{O}(\ell)$ time, and then they prove that only $\tilde{O}(\ell)$ items from S are mapped into the same memory module. (A Partial ℓ -relation routing

is the routing where at the most ℓ packets originate from any node and at the most ℓ packets are destined for any node.)

4.3 PRAM Emulation on the Mesh

Although Ranade's emulation technique [15] implies an asymptotically optimal algorithm for emulating a PRAM on the mesh, the underlying constant in the time bound is very high, say > 100 . In [12] Palis et al. present a better emulation algorithm whose time bound is only $4n + \tilde{o}(n)$.

As others, the emulation algorithm consists of mapping the shared memory locations of the PRAM on to the n^2 memory modules of the mesh using the universal hashing function introduced above. After the address mapping, the emulation problem reduces to two phases of routing. In both the phases the same routing algorithm was used. The same routing algorithm as the one given in [7] is used with a different analysis. Each phase will be finished in $2n + \tilde{o}(n)$ steps. Using the fact that if N items are mapped into N buckets, each bucket will get $\tilde{O}\left(\frac{\log N}{\log \log N}\right)$ items, it can prove a queue size of $O(\log n)$. Then with the emulation algorithm, the following theorem holds:

Each instruction of an EREW PRAM can be emulated on the mesh in $4n + \tilde{o}(n)$ steps. The queue size of the processors is $\tilde{O}(\log n)$.

The queue size of the algorithm can be reduced to $O(1)$ making use of the fact that if N items are mapped into N buckets, and if S is a collection of $\log N$ buckets, the number of items mapped into S will be $\tilde{O}(\log N)$. The improvement parallels the $2n + \tilde{O}(\log n)$ routing time algorithm presented in [7], with a slightly different analysis.

5 Conclusions

Valiant's two phase routing scheme has been proved to be a powerful technique for packet routing. It is also shown that making use of generating functions to handle random variables can simplify the analysis of the behavior of the routing algorithm and can also lead to a tighter upper bound. In particular, optimal randomized algorithms are also introduced for packet routing on networks with sub-logarithmic diameter. We have also surveyed optimal algorithms for emulating a PRAM on more realistic machine models. The model considered was a leveled network with sub-logarithmic diameter. We also introduced a $4n + \tilde{o}(n)$ steps emulation algorithm for an $n \times n$ mesh.

References

1. Akers, S., Harel, D. and Krishnamurthy, B.: The Star Graph: An Attractive Alternative to the n-Cube. *Proc. International Conference of Parallel Processing*, 1987, pp. 393-400.
2. Aleliunas, R.: Randomized parallel communication. *Proc. Symposium on Principles of Distributed Computing*, 1982, pp. 60-72.

3. Borodin, A. and J. E. Hopcroft: Routing, merging and sorting on parallel models of computation. *Proc. Symposium on Theory of Computing*, 1982, pp. 338-344.
4. Hoare, C.A.R.: Quicksort. *Computer Journal*, vol. 5, no. 1, 1962, pp. 10-15.
5. Karp, R. and Ramachandran, V.: Parallel Algorithms for Shared-Memory Machines. in *Handbook of Theoretical Computer Science*, North-Holland, 1990.
6. Karlin, A. and Upfal, E.: Parallel Hashing—An Efficient Implementation of Shared Memory. *Proc. Symposium on Theory of Computing*, 1986, pp. 160-168.
7. Krizanc, D., Rajasekaran, S., and Tsantilas, T.: Optimal Routing Algorithms for Mesh-Connected Processor Arrays. *Proc. Aegean Workshop on Computing*, 1988. Springer-Verlag Lecture Notes in Computer Science # 319, pp. 411-422.
8. Kunde, M., 'Routing and Sorting on Mesh-Connected Arrays,' *Proc. Aegean Workshop on Computing*, 1988. Springer-Verlag Lecture Notes in Computer Science # 319, pp. 423-433.
9. Leighton, T., Maggs, B., and Rao, S.: Universal packet routing algorithms. *Proc. Symposium on Foundations of Computer Science*, 1988, pp. 256-269.
10. Leighton, T., Makedon, F., and Tollis, I.G.: A $2n-2$ Step Algorithm for Routing in an $n \times n$ Array With Constant Size Queues. *Proc. Symposium on Parallel Algorithms and Architectures*, 1989, pp. 328-335.
11. Palis, M., Rajasekaran, S., and Wei, D.S.L.: Packet Routing and PRAM Emulation on Star Graphs and Leveled Networks. *Journal of Parallel and Distributed Computing*, vol. 20, no. 2, Feb. 1994, pp. 145-157.
12. Palis, M., Rajasekaran, S., and Wei, D.S.L.: Emulation of PRAMs on Leveled Networks. *20th International Conference on Parallel Processing*, Chicago, August, 1991, pp.I-418-421.
13. Pippenger, N.: Parallel communication with limited buffers. *Proc. Symposium on Foundations of Computer Science*, 1984, pp.127-136.
14. Rabin, M.O.: Probabilistic Algorithms. in:Traub, J.F., ed., *Algorithms and Complexity*, Academic Press, New York, 1976, pp. 21-36.
15. Ranade, A.G.: How to Emulate Shared Memory. *Proc. Symposium on Foundations of Computer Science*, 1987, pp. 185-194.
16. Solovay, R. and Strassen, V.: A Fast Monte-Carlo Test for Primality. *SIAM Journal on Computing*, vol. 6, 1977, pp. 84-85.
17. Upfal, E.: Efficient schemes for parallel communication. *Journal of the ACM*, vol.31, no.3, 1984, pp. 507-517.
18. Valiant, L.G.: A Scheme for Fast Parallel Communication. *SIAM Journal on Computing*, 11(2), 1982, pp. 350-361.
19. Valiant, L.G., Brebner, G.J.: Universal Schemes for Parallel Communication. *Proc. Symposium on Theory of Computing*, 1981, pp. 263-277.
20. Wei, D.S.L.: Fast Parallel Routing and Computation on Interconnection Networks. Ph.D. Thesis, Univ. of Pennsylvania, Jan. 1991.