

A Randomized Algorithm for Multipacket Routing on the Mesh¹

Sanguthevar Rajasekaran²

Department of CIS, Univ. of Pennsylvania
Philadelphia, PA 19104

Mukund Raghavachari

Computer Science Department
Princeton University, Princeton, NJ

¹A preliminary version of this paper was presented in the Symposium on Parallel and Distributed Processing, Dallas, Texas, Dec. 1991.

²Author's current address: Dept. of CIS, University of Florida, Gainesville, FL 32611

Multipacket Routing on the Mesh

Sanguthevar Rajasekaran

Department of CIS, Univ. of Pennsylvania

Philadelphia, PA 19104

(215) 898 0375

Abstract. In this paper we present a randomized algorithm for the multipacket routing problem on an $n \times n$ mesh. The algorithm completes with high probability in at the most $kn + o(kn)$ parallel communication steps, with a queue size of $k + o(k)$. The previous best known algorithm [3] takes $\frac{5}{4}kn + O(\frac{kn}{f(n)})$ steps with a queue size of $O(k f(n))$ (for any $1 \leq f(n) \leq n$). The algorithm that we will present is optimal with respect to queue size. The time bound is within a factor of two of the known lower bound.

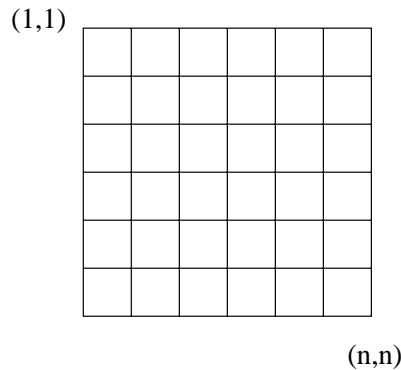


Figure 1: An $n \times n$ Mesh

1 Introduction

An important consideration in the design of any parallel algorithm is the amount of time that will be spent routing packets of information among the processors. Therefore, the development of fast and efficient packet routing algorithms is important. Lot of the past efforts in this area have focussed on mesh connected processor arrays due to their simple geometry and their practicality.

An $n \times n$ mesh connected processor array is composed of n^2 processors arranged in a square $n \times n$ grid, with no wraparound connections. The grid edges are the communication links between the processors and are bidirectional (see figure 1). Each processor can communicate with all its nearest neighbors in one time step. The control structure for the processors is assumed to be MIMD.

1.1 The Routing Problem

The problem of packet routing in a network is this: Each node in the network has a packet of information that has to be sent to some other node. The task is to send all the packets to their correct destinations quickly such that at the most one packet passes through any wire at any time. A special case of the routing problem is called *partial permutation routing*. In partial permutation routing, each node is the origin of no more than one packet and each node is the destination of at the most one packet. The *run time* of a packet routing algorithm is defined to be the time taken by the last packet to reach its destination, and the *queue*

size is defined to be the maximum number of packets any processor (or node) will have to store at any time during routing. Contentions for the edges can be resolved using a *priority scheme*. The ones we assume in this paper are the furthest destination first and the furthest origin first schemes. The routing algorithm specifies what path each packet should take and how to resolve contentions for the same edge.

1.2 Known and New Results

Most of the past results deal with the partial permutation routing problems (see e.g., [10] [8] [9] [2] [5] [6]). This paper deals with the problem of multipacket routing (also called the ‘ $k - k$ routing’ problem). In $k - k$ routing, at the most k packets originate from any node and at most k packets are destined for any node. It need not be the case that if one of the packets originating from a node (say i) is destined for a node (say j), then the other packets originating from i will also be destined for j . Kunde and Tensi [3] have shown that the $k - k$ routing problem can be solved in $\frac{5}{4}kn + O(\frac{kn}{f(n)})$ steps using a queue size of $O(kf(n))$ (for any $1 \leq f(n) \leq n$). They have also shown that for the special case of routing a sequence of k permutations, the time bound can be improved to $kn + O(\frac{kn}{f(n)})$, the queue size being the same. In this paper we present a $kn + o(kn)$ time, $k + o(k)$ queue size randomized algorithm for the general $k - k$ routing problem.

2 Preliminaries

2.1 The Queue Line Lemma

In the process of routing in a network, the time taken by any packet to reach its destination is dictated by two factors: 1) the *distance* between the packet’s origin and destination, and 2) the number of steps (also called the *delay*) the packet waits in queues. Valiant and Brebner proved a lemma known as the *Queue Line Lemma* [10] that enables one to compute an upper bound on the delay of any packet.

Consider the set of paths \mathcal{P} taken by the packets. Two packets are said to *overlap* if they share at least one edge in their paths. The set of paths is said to be *nonrepeating* if for any two paths in \mathcal{P} , the following statement holds: If these two paths meet, share some successive edges, and diverge, then they will never meet again.

The following lemma is proven in [10]:

Lemma 2.1 *The amount of delay any packet q suffers waiting in queues is no more than the number of packets that overlap with q , provided the set of paths taken by packets is nonrepeating.*

2.2 Chernoff Bounds

Let $X = B(n, p)$ stand for the number of heads in n independent flips of a coin, the probability of a head in a single flip being p . The following three facts (known as Chernoff bounds) are now folklore:

$$\text{Prob.}[X \geq m] \leq \left(\frac{np}{m}\right)^m e^{m-np}, \quad (1)$$

$$\text{Prob.}[X \geq (1 + \epsilon)np] \leq \exp(-\epsilon^2 np/2), \text{ and} \quad (2)$$

$$\text{Prob.}[X \leq (1 - \epsilon)np] \leq \exp(-\epsilon^2 np/3), \quad (3)$$

for any $0 < \epsilon < 1$, and $m > np$. By *high probability*, we mean a probability of $\geq (1 - n^{-\alpha})$ for any constant $\alpha \geq 1$. Let ‘w.h.p.’ stand for ‘with high probability’. We say a function $f(n)$ is $\tilde{O}(g(n))$ if $f(n)$ is $\leq c\alpha g(n)$ with probability $\geq (1 - n^{-\alpha})$ for any α and some constant c . Similar definitions can be given for other functions such as $\tilde{o}(\cdot)$.

3 $k - k$ Routing

3.1 $k - k$ Routing on a Linear Array

In this section we present an optimal algorithm for $k - k$ routing on a linear array.

Lemma 3.1 *$k - k$ routing can be performed on a linear array of n processors in $\frac{nk}{2}$ steps.*

Proof. This proof is similar to a proof given in [8]. Each packet travels along the shortest path from its origin to destination. The priority scheme employed is the furthest destination first scheme.

Consider a packet q that originates in node i ($1 \leq i \leq n$) and whose destination is j . Since the links are bidirectional, flow of packets in one direction does not affect the flow in

the other direction. Thus w.l.o.g. assume that j is to the right of i and all the packets are traversing from left to right.

Packet q can possibly be delayed by at the most $k(n - j)$ other packets (with higher priority). Also realize that q can only be delayed by $k(j - 1)$ other packets. Therefore, the number of **distinct** packets that can delay q is $\min\{k(n - j), k(j - 1)\}$. If q were not delayed by any other packet, it needs only $(j - i)$ steps to reach its destination. Therefore, applying the queue line lemma (lemma 2.1), the time needed for q to reach its destination is no more than $\min\{(n - j)k, (j - 1)k\} + (j - i)$. The maximum of this quantity over all possible i 's and j 's is $\frac{nk}{2}$. \square

Lemma 3.2 *If there are m packets on an n -node linear array with zero or more packets originating from any node and zero or more packets destined for any node, routing can be performed within $m + n - 1$ steps.*

Proof. This lemma is an immediate consequence of the queue line lemma. \square

The following lemma has been proven in [3]. Its proof is similar to that of lemma 3.1:

Lemma 3.3 *Let there be xn packets in a linear array. If for any j , the number of packets with an address $> j - 1$ is $\leq (n - j + 1)x + g(n)$, and the number of packets with an address $< j$ is $\leq (j - 1)x + g(n)$, then routing can be completed within $xn + g(n)$ steps using the furthest destination first priority scheme.*

3.2 Algorithm for $n \times n$ mesh

The algorithm that we shall present for $k - k$ routing on a mesh resembles the algorithm presented in [9]. We will first describe a $2kn + \tilde{o}(kn)$ algorithm which will be modified to run in $kn + \tilde{o}(kn)$ steps.

The grid is divided into two regions **S** and **I**, and packets with origins in these two regions are called *superior* packets and *inferior* packets respectively (see figure 2).

Inferior packets are routed using the following algorithm. The rows are divided up into $1/\epsilon$ strips of ϵn rows each. The algorithm has three phases. A packet at processor (x, y) , destined for processor (r, s) , is first routed along the column y to (w, y) , a processor chosen at random in the same column and strip as (x, y) . The packet is then sent to (w, s) along row w , after which, it is routed to its destination along column s .

The superior packets are routed using a slightly different algorithm. A processor (x, y) in the upper two **S** squares, with destination (r, s) sends its packet to (w, y) along column y ,

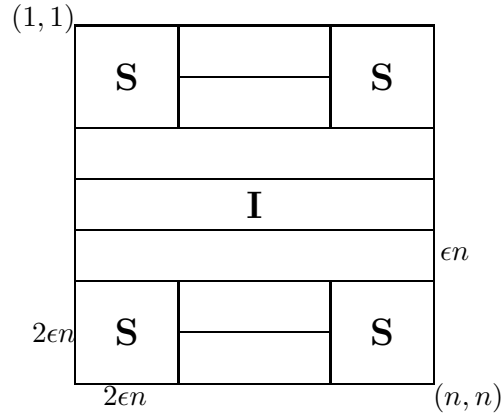


Figure 2: Inferior and Superior packets

where w is chosen at random from $\{2\epsilon n + 1, 2\epsilon n + 2, \dots, 2\epsilon n + (1/4)n\}$. The packet is then sent to (w, s) along row w , and then, to (r, s) along column s . ϵ is chosen to be less than $1/8$, so that superior packets in the upper half remain in the upper half after the randomization phase. The algorithm for the packets in the lower two S squares is symmetric.

During the first and third phases, no distinction is made between superior and inferior packets with respect to the queuing disciplines. If they contend for the same edge, a packet performing in its first phase takes precedence over one in its third phase. In phase II, superior packets take precedence over inferior packets and among the inferior (superior) packets the furthest origin first priority scheme is used. In phase III, packets that have further to go have higher priority.

3.3 Routing Time Analysis

Superior Packets

A superior packet q starting its phase II at (u, v) can be delayed by at the most $2\epsilon knv$ packets, each with probability $4/n$. Therefore, the number of packets that can potentially delay q is $m = B(2\epsilon knv, \frac{4}{n})$. Using Chernoff bounds (section 2.2, equation 2), we can show that m is $kv + \tilde{O}(k \log n)$. q needs to traverse a distance of $\leq n - v$ in phase II. Thus, superior packets will complete phase II in at the most $kn + \tilde{O}(k \log n)$ steps. Superior packets will complete phases I and III in at the most kn steps. Therefore, all superior packets will complete all three phases in $2kn + \tilde{O}(k \log n)$ steps.

Inferior Packets

For an inferior packet that starts phase II at row w , we have two possible cases.

1. $w \leq 2\epsilon n$ or $w \geq n - 2\epsilon n$

Suppose an inferior packet q starts phase II at row w , $w \leq 2\epsilon n$, and w.l.o.g., it moves from left to right. Suppose, the packet starts the phase at column t . Then w.h.p., the delay q suffers will be at the most $k(t - 2\epsilon n) + kn^\delta$, for some $\delta < 1$. (This follows from the fact that we use the furthest origin first priority scheme in this phase, and that the number of inferior packets that can delay q in phase II is $B(k(t - 2\epsilon n)\epsilon n, \frac{1}{\epsilon n})$, and an application of the Chernoff bounds equation 2). Since q has to travel a distance of no more than $n - t - 2\epsilon n$, it will complete the phase II in $\leq kn - 2k\epsilon n + kn^\delta$ steps w.h.p. Since the packet spends $\leq k\epsilon n$ steps in phase I and at the most kn steps in phase III (from lemma 3.2), it takes no more than $2kn - k\epsilon n + kn^\delta$ steps to complete all phases. This is $\leq 2kn$ for appropriate ϵ .

2. $2\epsilon n < w < n - 2\epsilon n$

For such a packet q , phase I completes in $k\epsilon n$ steps, and phase III in $kn - 2k\epsilon n$ steps. Suppose a packet starts phase II in column t . The number of inferior packets that delay q is $kt + kn^\delta$ (for some $\delta < 1$), w.h.p. The expected number of superior packets that will delay the packet during phase II is $16k\epsilon^2 n$. Using Chernoff bounds equation 2, we can show that the number of superior packets delaying our packet is no more than $k\alpha\epsilon^2 n$, w.h.p., for some $\alpha > 16$. Thus, the total routing time of the packet, in this case, is $k\epsilon n + kn - 2k\epsilon n + kn + kn^\delta + k\alpha\epsilon^2 n \leq 2kn$, for small enough ϵ .

Therefore, all inferior packets will complete all three phases in at the most $2kn$ steps w.h.p. (see also [9].)

3.4 Modification to the Algorithm

We can reduce the number of steps taken by the algorithm by making the following modifications. Initially, each inferior processor colors its packets black or white randomly. A packet has equal probability of being colored black or white. The mesh is partitioned into both vertical and horizontal slices of ϵn columns and rows respectively.

In phase I, a white packet chooses a random node in the same column and horizontal slice as its origin and goes there along its column of origin. Also in phase I, a black packet chooses a random node in the same row and vertical slice as its origin and goes there along

the row of origin. During phase II, all white packets are routed along rows till they reach their column destinations, while black packets are routed along columns till they reach their row destinations. In phase III, white packets are routed along columns to their destinations, while black packets are routed along rows. There is no change in the algorithm for superior packets.

It is likely that white and black inferior packets contend for the same edge. For instance, a white inferior packet in phase I may compete for an edge with a black inferior packet performing its phase II. Whenever there is such a conflict between black and white packets, preference is given to packets in lower phases. In the above example, the white packet will be given priority.

Theorem 3.1 *Using a randomized coloring scheme, routing can be performed in $kn + \tilde{o}(kn)$ steps on an $n \times n$ mesh with a queue size of $k + \tilde{o}(k)$.*

Proof. As a result of the coloring, the number of inferior packets that will perform their phase II along any row(column) and the number of inferior packets that will perform their phase III along any column(row) is no more than $kn/2 + kn^{3/4}$, w.h.p. This is due to the fact that the above number is a binomial, $B(kn, 1/2)$. Using analysis similar to that shown in the previous section, we find that, if we use a randomized coloring scheme, routing can be completed in $kn + \tilde{o}(kn)$ steps.

The conflict between white and black packets does not affect the time bound for the following reason: Consider the example of a white packet in phase I conflicting with a black packet in phase II. If such a conflict occurs, it means that the black packet has completed its phase I well within ϵkn steps, and even if it waits for all the (black and white) packets to complete their phase I, it will start its phase II at the latest by step ϵkn , thus unaffected the analysis. Conflicts of other kinds can also be argued similarly. The queue size is proved next. \square

Queue Size Analysis

The queue size of the above algorithm in any phase is readily seen to be no more than the queue size at the beginning of the phase. For example in phase I, the number of packets that will end up in any node is upper bounded by $B(k\epsilon n, \frac{1}{\epsilon n})$. The expected value of this number is k . Using Chernoff bounds (equation 1), this number is $O(k \log n)$ with high probability. In a similar way we also see that the queue sizes of phase II and phase III are $O(k \log n)$ packets with high probability.

Using ideas similar to ones given in [9], we can reduce the queue size to $k + \tilde{o}(k)$ packets. The crucial fact used is that the expected queue size at any node in any phase is k . This in turn means that the total queue size of any n^ϵ successive nodes in the mesh is $kn^\epsilon + \tilde{O}(\sqrt{kn^\epsilon \log n})$ (for some fixed $\epsilon < 1$).

The idea is to partition each column (as well as row) into slices of n^ϵ successive nodes. At any time in the algorithm, packets that have to be stored in each such slice are uniformly distributed among the nodes in the slice. That is, if a node in a slice has to store more than $k + c\sqrt{\frac{k \log n}{n^\epsilon}}$ packets (for some constant $c > 1$), it will send the additional packets to its neighbor in the slice, and the neighbor will do the same thing. With high probability, this redistribution will be local to each slice. Any packet that gets redistributed in its slice can only suffer an additional delay of $\tilde{O}(kn^\epsilon)$.

4 Conclusion

We've presented a routing algorithm for $k - k$ routing that is optimal with respect to queue size. The known lower bound for $k - k$ routing is $\frac{kn}{2}$ [3]. Our algorithm has a time bound that is within a factor of 2 of this lower bound.

Postscript

Recently, Kunde [4] has presented a deterministic algorithm for the general $k - k$ routing whose time bound is $kn + o(kn)$ and queue size is k . Also randomized algorithms for $k - k$ routing and cut through routing whose time bounds very nearly match the lower bound have been given in [1].

References

- [1] M. Kaufmann, S. Rajasekaran, and J. Sibeyn, Matching the Bisection Bound for Routing and Sorting on the Mesh, in Proc. ACM Symposium on Parallel Algorithms and Architectures, 1992, pp. 31-40.
- [2] M. Kunde, Routing and Sorting on Mesh-Connected Processor Arrays, in Proc. VLSI Algorithms and Architectures: AWOC 1988. Springer-Verlag Lecture Notes in Computer Science #319, Springer-Verlag, pp. 423-33.

- [3] M. Kunde and T. Tensi, $(k - k)$ Routing on Multidimensional Mesh-Connected Arrays, *Journal of Parallel and Distributed Computing* 11, 1991, pp. 146-155.
- [4] M. Kunde, Concentrated Regular Data Streams on Grids: Sorting and Routing Near to the Bisection Bound, in *Proc. IEEE Symposium on Foundations of Computer Science*, 1991.
- [5] T. Leighton, F. Makedon, and I.G. Tollis, A $2n - 2$ Step Algorithm for Routing in an $n \times n$ Array With Constant Size Queues, in *Proc. ACM Symposium on Parallel Algorithms and Architectures*, 1989, pp. 328-35. To appear in *Algorithmica*.
- [6] S. Rajasekaran and R. Overholt, Constant Queue Routing on a Mesh, *Proc. Symposium on Theoretical Aspects of Computer Science*, 1990. Springer-Verlag Lecture Notes in Computer Science #480, pp. 444-455. Also in *Journal of Parallel and Distributed Computing* 15, 1992, pp. 160-166.
- [7] S. Rajasekaran and M. Raghavachari, Optimal Randomized Algorithms for Multipacket and Cut Through Routing on the Mesh, in *Proc. IEEE Symposium on Parallel and Distributed Processing*, 1991, pp. 305-311.
- [8] S. Rajasekaran and T. Tsantilas, An Optimal Randomized Routing Algorithm for the Mesh and A Class of Efficient Mesh-like Routing Networks, *Proc. 7th Conference on Foundations of Software Technology and Theoretical Computer Science*, 1987. Springer-Verlag Lecture Notes in Computer Science #287, pp. 226-241.
- [9] S. Rajasekaran and T. Tsantilas, Optimal Routing Algorithms for Mesh-Connected Processor Arrays, *Algorithmica* 8, 1992, pp. 21-38.
- [10] L.G. Valiant, A scheme for fast parallel communication, *SIAM J. on Computing*, 11:2, 1982, pp. 350-361.

Biographies

Sanguthevar Rajasekaran received his M.E. degree in Automation from the Indian Institute of Science (Bangalore) in 1983, and his Ph.D. degree in Computer Science from Harvard University in 1988. From 1988 to 1994 he was employed in the Computer and Information Sciences Department of the University of Pennsylvania as an Assistant Professor. Currently, he is an Associate Professor in the Department of Computer and Information Sciences of the University of Florida. His research interests include Parallel Algorithms, Randomized Computing, Combinatorial Optimization, Learning Theory, Animation, Simulation, etc.

Mukund Raghavachari received his B.S. degree from the Department of Computer and Information Sciences of the University of Pennsylvania in May 1991. Since then he is a graduate student in the Computer Science Department of Princeton University. His research interests include Parallel Algorithms and Parallel Systems.