

Optimal Randomized Algorithms for Multipacket and Cut Through Routing on the Mesh

Sanguthevar Rajasekaran

Department of CIS
Univ. of Pennsylvania
Philadelphia, PA 19104

Mukund Raghavachari
Computer Science Department
Princeton University
Princeton, NJ

Abstract

In this paper, we present a randomized algorithm for the multipacket (i.e., $k - k$) routing problem on an $n \times n$ mesh. The algorithm completes with high probability in at the most $kn + O(k \log n)$ parallel communication steps, with a constant queue size of $O(k)$. 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 $1 \leq f(n) \leq n$). We will also present a randomized algorithm for the cut through with partial cuts model permutation routing problem for the mesh that completes in at the most $kn + O(k \log n)$ steps, with a constant queue size of $O(k)$, where k is the number of flits that each packet is divided into. The previous best result [6] was also randomized and had a time bound of $kn + O(\frac{kn}{f(n)})$ with a queue size of $O(k f(n))$ (for any $1 \leq f(n) \leq n$). The two algorithms that we will present are optimal with respect to queue size. The time bounds are within a factor of two of the only known lower bound.

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. Each processor can communicate with all its nearest neighbors in one time step. The

control structure for the processors is assumed to be MIMD.

Most of the past results in this area have been derived for the Store and Forward model of packet routing. In this model, the packet is considered as the atomic element of routing, i.e., one whole packet can be transmitted along a link in one time step. However, since the width of the communication links is usually smaller than the size of the packets, it is more convenient (and practical) to break up each packet into smaller pieces, called flits.

If each packet is broken up into k flits, where k depends on the width of the channel, the problem can be studied under two different approaches. We can consider the k flits to be k distinct packets, which are routed independently. This is known as the multipacket routing approach [3].

Alternatively, we can consider the k flits as a *snake*. All flits follow the first one, known as the head, to the destination. A snake may never be broken, i.e., at any given time, consecutive flits of a snake are at the same or adjacent processors. In the ‘cut through with partial cuts’ (referred to from here on simply as ‘cut through’) routing, at any given time, there can be at the most $n/2$ processors that hold more than one flits of a snake [6].

Makedon & Simvonis [6] and Kunde & Tensi [3] have presented efficient algorithms for the cut through model routing and multipacket routing problems, respectively. A large number of algorithms have been designed for the Store and Forward model [14] [10] [11] [2] [5] [9]. [6] have presented a deterministic algorithm for cut through permutation routing that completes in $\frac{3}{2}kn + O(\frac{kn}{f(n)})$ steps with a queue size of $O(k f(n))$ (for any $1 \leq f(n) \leq n$). They have also presented a randomized algorithm that takes at the most $kn + O(\frac{kn}{f(n)})$ steps with a queue size of $O(k f(n))$. For the multipacket routing problem, [3] have presented a deterministic algorithm for $k - k$ routing that takes at the

most $\frac{5}{4}kn + O(\frac{kn}{f(n)})$ steps to complete with a queue size of $O(k f(n))$ (for any $1 \leq f(n) \leq n$).

In Section 3 of this paper, we will present a randomized algorithm for cut through routing that completes in $kn + O(k \log n)$ steps with a constant queue size, i.e., $O(k)$ flits. In Section 4, we will show that multipacket (i.e., $k - k$) routing can be performed in $kn + O(k \log n)$ steps with a constant queue size of $O(k)$ using a randomized algorithm.

2 Preliminaries

2.1 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 *permutation routing*. In permutation routing, each node is the origin of exactly one packet and each node is the destination of exactly 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.

2.2 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* [14] 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 [14]:

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.*

2.3 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:

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

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

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

for any $0 < \epsilon < 1$, and $m > np$. By *high probability*, we mean a probability of $\geq (1 - n^{-\alpha})$ for any constant $\alpha \geq 1$.

3 Cut Through Routing

3.1 Cut Through Routing on a Linear Array

In any network, since the width of the communication links is usually less than the size of a packet, it is more practical to break up each packet into k flits, where k depends on the width of the link. Though the Store and Forward model has drawn most of the attention in the past, it is the cut through model that is more commonly used in practice [6].

Formally, a snake of k flits, s , at any time t , is defined by a $(k + 1)$ -tuple, $(s_1, s_2, \dots, s_i, t)$, where s_1, s_2, \dots, s_i , $i \leq k$, are consecutive processors that contain at least one flit of the snake. The *length* of a snake s at time t , $length(s, t)$, is defined to be the number of processors over which the snake is distributed. A snake is in *full-extension* if $length(s, t) = k$. The function *collisions*(s, t) for a snake s , at time t , is defined to be the number of processors that hold more than one flit of snake s , at time t [6].

Lemma 3.1 *Permutation routing can be performed on a linear array of n processors in $\frac{n(k+1)}{2}$ steps under the cut through model. Here k is the number of flits in each packet.*

Proof. This lemma has already been proven by Makedon and Simvonis [6]. We give a slightly simpler proof here. The algorithm used by the processors is quite simple. At each time unit, every processor transmits the flit in the head of its queue. At the same time it receives a flit from each neighbor and appends these flits to appropriate queues.

The proof by Makedon and Simvonis was given on a more restricted model of routing called the ‘restricted cut through model’. An upper bound obtained under this model will clearly be an upper bound on the (more powerful) cut through model as well. In the restricted cut through model, once a packet gains full extension, it will not be compressed again. Transmission of a packet starts at time instances that are multiples of k .

An immediate consequence of the restricted cut through model and the algorithm described above is that at least one packet gains full extension every k steps. We make use of the furthest destination first priority 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 $(n - j)$ other packets (with higher priority). Also realize that q can only be delayed by $(j - 1)$ other packets. Therefore, the number of **distinct** packets that can delay q is $\min\{(n - j), (j - 1)\}$. If q were not delayed by any other packet, it needs only $k + (j - i)$ 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 $\min\{(n - j)k, (j - 1)k\} + k + (j - i)$. The maximum of this quantity over all possible i ’s and j ’s is $\frac{n(k+1)}{2} \square$

Using similar arguments we can also prove the following lemma:

Lemma 3.2 *If there are m packets on a linear array such that each processor has possibly more than one packets to start with and each processor is the destination of exactly one packet, routing can be completed in $(k - 1)m + n$ steps.*

3.2 Algorithm for $n \times n$ mesh

The algorithm that we shall present for cut through model routing on a mesh is for permutation routing and it resembles the algorithm presented in [11]. We will first describe a $2kn + O(k \log n)$ algorithm which will be modified to run in $kn + O(k \log n)$ steps, where k is the number of flits in a packet.

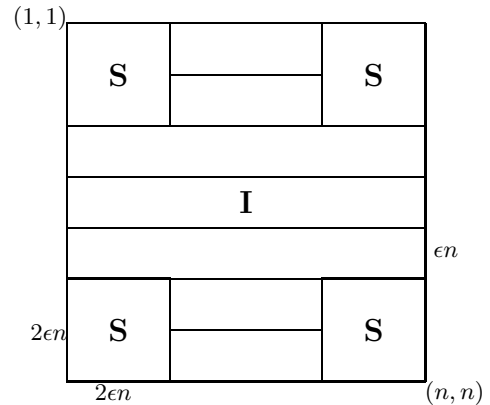


Figure 1: Inferior and Superior packets

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 1).

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 , 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 $2\epsilon nv$ packets, each with probability $4/n$. Therefore, the number of packets that can potentially delay q is $m = B(2\epsilon nv, \frac{4}{n})$. Using Chernoff bounds (section 2.1, equation 2), we can show that m is $v + O(\log n)$, with high probability. q needs to traverse a distance of $\leq n - v$ in phase II. Thus, superior packets will complete phase II in at the most $kn + O(k \log n)$ steps, with high probability. Superior packets will complete phases I and III in at the most kn steps. Therefore, all superior packets will complete all three phases in $2kn + O(k \log n)$ steps with high probability.

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 with high probability, the delay q suffers will be at the most $kt + kn^\delta$, 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 any inferior packet in phase II is $B(t\epsilon n, \frac{1}{\epsilon n})$, 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 with high probability. 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 $t + n^\delta$ (for some $\delta < 1$), with high probability. The expected number of superior packets that will delay the packet during phase II is $32\epsilon^2 n$. Using Chernoff bounds equation 2, we can show that the number of packets delaying our packet is no more than $\alpha\epsilon^2 n$, with high probability, for some $\alpha > 32$. 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 with high probability (see also [11].)

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 flips a coin and colors its packet black or white depending on the result. The mesh is partitioned into both vertical and horizontal slices of ϵn columns and rows respectively.

In phase I, all the white packets choose a random node in the same column and horizontal slice as their origin and go there along the column of origin. Also in phase I, the black packets choose a random node in the same row and vertical slice as their origin and go there along the row of origin. During phase II, all white packets are routed along rows till they reach their column destination, while black packets are routed along columns till they reach their row destination. 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 $kn + O(k \log n)$ steps on an $n \times n$ mesh with queue size of $O(k)$ flits, with high probability.*

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 $n/2 + n^{3/4}$, with high probability. This is due to the fact that the above number is a binomial, $B(n, 1/2)$. 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 $kn + O(k \log n)$ steps with high probability.

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 by the analysis. Conflicts of other kinds can also be argued similarly. \square

Queue Size Analysis

The queue size of the above algorithm in any phase is 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 $B(\epsilon n, \frac{1}{\epsilon n})$. Using Chernoff bounds (equation 2), this number is $O(\log n)$ with high probability. In a similar way we also see that the queue sizes of phase II and phase III are $O(\log n)$ packets (i.e., $O(k \log n)$ flits) with high probability. Using ideas similar to ones given in [11], we can reduce the queue size to $O(k)$ flits (i.e., a constant number of packets). The crucial fact used is that the queue size of any $\log n$ successive processors in the array is still $O(k \log n)$, with high probability. Each column (as well as row) is partitioned into slices of $\log n$ successive nodes. Packets that have to be stored in each such slice are distributed among the nodes in the slice. That is, if a node in a slice has to store more than ck flits (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.

4 Multipacket Routing

4.1 Preliminaries

In Multipacket routing, each packet is broken up into k flits, and these flits behave as though they are independent entities. In particular, each flit carries along with it, its source and destination numbers. Elementary calculations show that breaking up a packet into k flits results in a speedup by a factor of k in transportation time [3]. The problem of $k-k$ routing is the problem of routing where exactly k packets originate from any node and exactly k packets are destined for any node. It need not be the case that if one of the k packets originating from a node (say i) is destined for a node (say j), then the other $k-1$ 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 section we present a $kn + O(k \log n)$ time, $O(k)$ queue size randomized algorithm for the general $k-k$ routing problem.

Lemma 4.1 *$k-k$ routing can be completed on an n -node linear array in $\frac{n(k+1)}{2}$ steps under the multipacket model.*

Proof. is very similar to that of Lemma 3.1 and has been already proven in [10] and [3] \square .

Lemma 4.2 *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. An immediate consequence of the queue line lemma. \square

The following lemma has been proven in [3] the proof of which is similar to that of lemma 4.1:

Lemma 4.3 *Let there be xn packets in a linear array. If 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.*

4.2 Algorithm

The algorithm for Multipacket routing is similar to the one presented in the last section. There are k packets initially at each node. Each processor flips a 2-sided coin k times and colors its packets black or white depending on the outcomes. The packets are then routed in exactly the same way as they are for the cut through model, except now that there are no flits but only independent packets. Also, now the superior packets are also colored black or white depending on the outcomes of coin flips. White and black superior packets execute symmetric but opposite algorithms (i.e., in phase I, a white packet chooses a random node in the column of its origin and goes there, whereas a black packet chooses a random node in the row of its origin and goes there, and so on.) Conflicts between white and black packets are resolved by assigning higher priority to packets in lower phases.

Theorem 4.1 *Using this algorithm, $k-k$ routing will take $kn + O(k \log n)$ steps, on an $n \times n$ mesh with queue size of $O(k)$, with high probability.*

Proof: Lemma 4.3 is crucial to the proof. The analysis is similar to the one in section 3. A superior packet completes its phases I and III in no more than $kn/2$ steps. In phase II, a superior packet that starts from column v can only be delayed by $8.5\epsilon kv + O(k \log n)$ other packets with high probability. It needs to traverse a distance of at the most $n - v$. Put together, the time needed for a superior packet in phase II is no more than $8.5\epsilon kv + O(k \log n) + n - v$, which is $\leq \frac{kn}{2} + O(k \log n)$, for a proper ϵ . Thus a superior packet will complete all the three phases in $\leq kn + O(k \log n)$ steps with high probability.

An inferior packet spends at the most $k\epsilon n$ steps in phase I. In phase II, if an inferior white packet q starts from row w and if $w \leq 2\epsilon n$ or $w \geq n - 2\epsilon n$, it will complete phase II in at the most $\frac{kn}{2} - 2k\epsilon n + kn^\delta$ steps with high probability. Phase III can be completed in $\frac{kn}{2} + kn^\delta$ steps with high probability. This is because only $\frac{kn}{2} + kn^\delta$ packets will be performing their phase III along any column with high probability and an application of lemma 4.3. Thus q needs no more than $kn - k\epsilon n + O(kn^\delta)$ steps for all the three phases. This number of steps is $\leq kn$ for appropriate ϵ .

The case of a white inferior packet starting from a row w such that $2\epsilon n < w < n - 2\epsilon n$ is similar. The same analysis applies to the black inferior packets as well. The time bound remains the same even after accounting for conflicts between white and black packets. The reason is, if a packet in a lower phase conflicts with a packet in a higher phase, priority is given to the packet in the lower phase. If at all such a conflict occurs, it implies that the packet in higher phase has completed its lower phases well ahead of time and hence even if it is delayed by the lower priority packet, it will reach its destination within the stated time. (See also section 3.4) \square

5 Conclusion

We've presented routing algorithms that are optimal with respect to queue size. It remains an open question if the time bound of the algorithms for both multipacket and cut through routing is optimal. The only known lower bound is for the multipacket routing and is $\frac{kn}{2}$ [3]. There is a large gap between the best known upper and lower bounds. Improving the bounds is a possible future area of research.

Postscript

Recently, Kunde [4] has announced to present a deterministic algorithm for the general $k-k$ routing whose time bound is $kn + o(kn)$ and queue size is k . But no such result has been proven for the cut through model of routing.

References

- [1] Dalley, W. 'Performance Analysis of k -ary n -cube Interconnection Networks,' Submitted to *IEEE Trans. on Computers*, 1988.
- [2] Kunde, M. 'Routing and Sorting on Mesh-Connected Processor Arrays,' Proc. VLSI Algorithms and Architectures: AWOC 1988. Springer-Verlag Lecture Notes in Computer Science #319, Springer-Verlag, pp. 423-33.
- [3] Kunde, M., and Tensi, T. 'Multi-Packet Routing on Mesh Connected Arrays,' Proc. ACM Symposium on Parallel Algorithms and Architectures, 1989, pp. 336-343.
- [4] Kunde, M., 'Concentrated Regular Data Streams on Grids: Sorting and Routing Near to the Bisection Bound,' to be presented in the IEEE Symposium on Foundations of Computer Science, 1991.
- [5] 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. ACM Symposium on Parallel Algorithms and Architectures, 1989, pp. 328-35.
- [6] Makedon, F., Simvonis, A. 'On bit Serial packet routing for the mesh and the torus.' Proc. third Symposium on Frontiers of Massively Parallel Computation, 1990, pp. 294-302.
- [7] Makedon, F., Simvonis, A. 'Fast Parallel Communication on Mesh Connected Machines with Low Buffer Requirements,' Proc. International Conference on Computer Design, 1990.
- [8] Ngai, J.Y. 'A Framework for Adaptive Routing in Multicomputer Networks,' Proc. Symposium on Parallel Algorithms and Architectures, 1989, pp. 1-9.
- [9] Rajasekaran, S., and Overholt, R. '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. To appear in JPDC.

- [10] Rajasekaran, S., and Tsantilas, T. '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.
- [11] Rajasekaran, S., and Tsantilas, T. 'Optimal Routing Algorithms for Mesh-Connected Processor Arrays,' Proc. VLSI Algorithms and Architectures: AWOC, 1988. Springer-Verlag Lecture Notes in Computer Science #319, pp. 411-22. To appear in Algorithmica.
- [12] Schnorr, C.P., Shamir, A. 'An Optimal Sorting Algorithm for Mesh Connected Computers,' Proc. 18th ACM Symposium on Theory of Computing, 1986, pp. 255-263.
- [13] Thompson, C., and Kung, H.T., 'Sorting on a Mesh-Connected Parallel Computer,' CACM, vol. 20, 1977, pp. 263-270.
- [14] Valiant, L.G. 'A scheme for fast parallel communication,' SIAM J. on Computing, 11:2, 1982, pp. 350-361.