

A COMPARISON OF MESHES WITH STATIC BUSES AND HALF-DUPLEX WRAP-AROUNDS

DANNY KRIZANC

*Department of Computer Science, University of Rochester
Rochester, NY 14627, U.S.A.*

and

SANGUTHEVAR RAJASEKARAN

*Department of Computer and Information Sciences, University of Pennsylvania
Philadelphia, PA 19104, U.S.A.*

and

SUNIL M. SHENDE

*Department of Computer Science and Engineering, University of Nebraska
Lincoln, NE 68588, U.S.A*

Received (received date)

Revised (revised date)

Communicated by (Name of Editor)

ABSTRACT

We investigate the relative computational powers of a mesh with static buses and a mesh with half-duplex wrap-arounds. The latter model is like a torus, except that any wrap-around link of the architecture can only transmit data in one of the two directions at any clock tick. We show that the permutation routing problem can be solved as efficiently on a linear array augmented with a half-duplex wrap-around link, as on a linear array with an augmented broadcast bus. We also present a routing algorithm for a two-dimensional (2D) mesh with half-duplex wrap-around links whose run time is close to that of the best known algorithm for routing on a 2D mesh with broadcast buses in each dimension. In addition, we show that on an $n \times n$ 2D mesh with broadcast buses, randomized sorting of n^2 elements can be accomplished in time that is only $o(n)$ more, with high probability, than the time needed for permutation routing.

Keywords: Mesh connected computer, Packet Routing, Sorting, Broadcast Bus, Randomized Algorithms

1. Introduction

Our basic computational model is the classical mesh, denoted M , both one-dimensional (a *linear array* with n processors) and two-dimensional (an $n \times n$ square grid with no wrap-around connections). A two-dimensional (henceforth, 2D) mesh

contains a processor at each grid point connected to its four (or less) neighbors in the grid via full-duplex links. In every time step, it is assumed that each processor can communicate with any number of its neighbors (this model is referred to as the MIMD model in the literature). The maximum storage required during any time step by a processor is called the *queue size* of the algorithm.

We consider two extensions of the classical mesh M . The mesh with static buses, denoted as M_b , is obtained from the classical mesh by augmenting the latter with exclusive-write, concurrent-read broadcast buses along each dimension. Thus, the linear array M_b contains a single bus that connects every processor in the array. The 2D mesh M_b contains $2n$ buses, one for each row and column of the mesh. Our second model is the classical mesh augmented with *half-duplex, wrap-around links*, and is denoted as M_w . This model is identical to the *torus*, except that the wrap-around links are constrained to be half-duplex, i.e. at any given clock tick, at most one of the endpoints of any such link is able to send a packet along the link.

We shall consider two main problems: permutation routing and sorting. These problems have been studied extensively for general r -dimensional meshes; see, e.g., [5, 11]. Extensions of the basic architecture including those with added broadcast buses have also been studied; see, e.g., [4, 13]. In the packet routing problem, a packet of information consists of some data (irrelevant to the problem) and the destination index (or indices) of some processor(s) in the architecture. In this paper, we shall mainly consider *permutation* routing, viz. there is at most one packet at each processor initially, and each processor may be the destination of no more than one packet. The problem of sorting on a mesh can be described as follows. There is a key at each node in the mesh, and the task is to rearrange the keys in ascending order according to some *indexing scheme*. The indexing scheme assumed in this paper is the blockwise snakelike row major indexing (also assumed in [2, 5, 9], for example).

The main motivation of this paper is to compare the performance of these models in the context of permutation routing, so as to gain some theoretical insight into the relative power of buses and half-duplex wrap-around links. We also describe a new sorting algorithm for the 2D model M_b .

2. Permutation Routing on M_b and M_w

It is well known that on the linear torus with a *full-duplex* wrap-around link, $\lfloor n/2 \rfloor$ time steps are necessary and sufficient to perform permutation routing. The linear array M_w is weaker than the torus. It also appears to be weaker than the linear array M_b , on which permutation routing can be performed in $2n/3$ steps, and $2n/3$ steps are needed [7]. However, we demonstrate the surprising result that both linear arrays M_w and M_b have *identical* permutation routing complexity.

Theorem 1 *Permutation routing on the linear array model M_w takes no more than $2n/3$ time steps, and $2n/3$ steps are necessary.*

Proof: Consider a linear array M_w and partition the n processors of M_w into three equal contiguous segments with $n/3$ processors in each segment. In particular, let A denote the segment of processors $\{i : 1 \leq i \leq n/3\}$, let B denote the

segment $\{i : n/3 + 1 \leq i \leq 2n/3\}$, and let C denote the remaining segment. For $X, Y \in \{A, B, C\}$, let X_Y denote the set of packets initially in segment X with destinations in segment Y .

For the lower bound, consider the (partial) permutation routing problem in which every A -processor i ($1 \leq i \leq n/3$) initially contains a single packet destined for the C -processor ($2n/3 + i$) in the array, and vice versa. Thus, there are $2n/3$ relevant packets, with each packet at a distance of $2n/3$ from its destination. If some packet *does not* use the half-duplex wrap-around link during the course of routing, it will need at least $2n/3$ time steps to reach its destination. On the other hand, if all the $2n/3$ packets use the wrap-around link, the half-duplex constraint on communication implies that at least $2n/3$ distinct time steps will be needed to deliver the packets across the link. Hence, at least $2n/3$ steps are necessary to route the permutation.

To demonstrate the upper bound, we describe a two-phase routing algorithm, with each phase executing for exactly $n/3$ time steps.

Phase I:

- C_A packets move to processor n , traverse the wrap-around link to processor 1, and then use the remaining time steps in phase I to progress towards their destinations in segment A.
- A_C packets undergo *rearrangement* within segment A. In particular, an A_C packet destined for processor $2n/3+k$ ($1 \leq k \leq n/3$) is sent to the intermediate processor k .
- The remaining packets proceed along shortest paths toward their destinations, without using the wrap-around link.

Phase II:

- C_A packets continue towards their destinations (now within segment A).
- The rearranged A_C packets move to processor 1, traverse the wrap-around link to processor n , and then proceed toward their destinations in segment C .
- A_B, B_A, B_C and C_B packets continue towards their respective destinations, with B_A packets yielding priority to the rearranged A_C packets in the event of edge contention.

Analysis: It is clear that the wrap-around link is only used in one direction in each phase, by C_A packets in phase I and by the rearranged A_C packets in phase II. Since every processor initially contains at most one packet, the queue size requirement is at most 3. Observe that all packets can move without delays in phase I. Hence, A_A, B_B and C_C packets complete their routing within the phase. Similarly, in phase II, A_C, C_A, A_B, B_C and C_B packets suffer no delays and hence reach their respective destinations by the end of phase II. Therefore, it only remains to show that every B_A packet that has not reached its destination by the end of phase I, completes its routing by the end of phase II.

Consider, without loss of generality, such a B_A packet which originates at processor $(n/3 + i)$ and is destined for processor j , for some $1 \leq j < i \leq n/3$. It

reaches processor i at the end of phase I without yet reaching its final destination. In phase II, the packet may be delayed by at most $(n/3 - i + 1)$ rearranged A_B packets to its “right” in segment A . Hence, after phase I, the packet finishes routing in $(n/3 - i + 1) + (i - j) = n/3 - (j - 1) \leq n/3$ additional time steps. \square

We now turn our attention to permutation routing on the 2D $n \times n$ meshes M_b and M_w . Some results for the former model are known, among them being a lower bound of $0.691 n$ steps [1], a simple three-phase algorithm that routes all permutations in $7n/6 + O(\frac{n}{q})$ steps with queue size of $O(q)$ for all q [7], and a more complicated routing algorithm with runtime $n + o(n)$ and queue size $o(n)$ [8].

The lower bound of $2n/3$ steps on the linear array M_w established earlier, can be easily extended to the 2D model M_w as well. However, the best upper bound for permutation routing on the 2D model M_w that we can derive is $3n/2 + O(\frac{n}{q})$ steps with queue size $O(q)$ for any q , $2 \leq q \leq n$. The routing algorithm is an adaptation of the three-phase algorithm for M_b (for details of the latter algorithm, see [7]). Omitting technical details, we can show that the same three phases can be implemented on M_w , respectively, in $O(\frac{n}{q})$ steps, $n/2$ steps, and n steps. The algorithm uses intermediate queues of size $O(q)$ to attain the stated time and queue size bounds. To our knowledge, this is the best known complexity result for routing on the 2D mesh M_w .

3. Randomized Sorting with Static Buses

We show here that sorting of n^2 elements can be accomplished on an $n \times n$ mesh with fixed buses in time that is only $o(n)$ more than the time needed for permutation routing with high probability (abbreviated as w.h.p. from hereon). If one employs the improved routing algorithm of Leung and Shende [8] the run time for sorting will be $n + O(\frac{n}{q})$ steps w.h.p., the queue size being $O(q)$ (for any $2 \leq q \leq n$).

Many optimal algorithms have been proposed in the literature for sorting on the conventional mesh (see e.g., [6]). A $2n + o(n)$ step randomized algorithm has been discovered for sorting by Kaklamani and Krizanc [2]. But $2n - 2$ is a lower bound for sorting on the conventional mesh. Recently Rajasekaran and McKendall [10] have presented an $n + o(n)$ randomized algorithm for routing on a reconfigurable mesh.

Our approach is based on *random sampling*, a vital technique used in the design of parallel algorithms for comparison problems (including sorting and selection). Reischuk’s [12] sorting algorithm is a good example. Given n keys, the idea is to: 1) randomly sample n^ϵ (for some constant $\epsilon < 1$) keys, 2) sort this sample (using any nonoptimal algorithm), 3) partition the input using the sorted sample as splitter keys, and 4) to sort each part separately in parallel. Similar ideas have been used in many other works as well (see e.g., [3, 2, 9, 10]).

Let $X = k_1, k_2, \dots, k_n$ be a given sequence of n keys and let $S = \{k'_1, k'_2, \dots, k'_s\}$ be a random sample of s keys (in sorted order) picked from X . X is partitioned into $(s + 1)$ parts defined as follows. $X_1 = \{\ell \in X : \ell \leq k'_1\}$, $X_j = \{\ell \in X : k'_{j-1} < \ell \leq k'_j\}$ for $2 \leq j \leq s$, and $X_{s+1} = \{\ell \in X : \ell > k'_s\}$. The following lemma [12] probabilistically bounds the size of each of these subsets, and will prove helpful to

our algorithm. (We say a function $f(n)$ is $\tilde{O}(g(n))$ if $f(n)$ is $\leq c\alpha g(n)$ for all large n and for some constant c with probability $\geq (1 - n^{-\alpha})$.)

Lemma 1 *The cardinality of each X_j ($1 \leq j \leq (s + 1)$) is $\tilde{O}(\frac{n}{s} \log n)$.*

Next we describe our algorithm and prove its time bound. This algorithm is similar to the one given in [10]. We only provide a brief summary of the algorithm. More details can be found in [3] or [10]. The mesh is partitioned into blocks of size $n^{4/5} \times n^{4/5}$.

- (i) A random sample of size very nearly $n^{3/5}$ is chosen and broadcast to the whole mesh, such that each block stores a copy of all the splitter keys.
- (ii) We compute the partial ranks of the sample keys in each block after sorting the block.
- (iii) Then we perform a prefix sums operation on these partial ranks so as to obtain the global ranks of the sample keys.
- (iv) Now we route each packet to an approximate destination that is a random node in an appropriate block of size $n^{4/5} \times n^{4/5}$. This approximate destination is very close to its actual destination and depends on the two splitter keys between which it falls. In particular, the approximate destination of any packet will be at most a block away from its actual destination w.h.p.
- (v) Next we sort the individual blocks and compute the rank of each key in the mesh.
- (vi) Finally we route the packets to their actual destinations.

Analysis: The key to the analysis is the observation that the global ranks of the sample keys can be computed in $o(n)$ steps. This observation was first made in [10] in connection with sorting on a reconfigurable mesh.

Step (i) takes $O(n^{3/5})$ steps, since a single key can be broadcast to the whole mesh in $O(1)$ steps using the buses. Step (ii) involves sorting blocks of size $n^{4/5} \times n^{4/5}$ (together with the sample keys) and can be completed in $O(n^{4/5})$ time using any standard sorting algorithm (such as Schnorr and Shamir's [6]). In step (iii), the global rank of a single key can be computed in time $O(n^{1/5})$. This can be done for instance by concentrating all the partial ranks of this key in a region of size $n^{1/5} \times n^{1/5}$. Thus the global ranks of all the keys can be determined in time $O(n^{1/5} \times n^{3/5}) = O(n^{4/5})$. In step (iv), routing takes $n + o(n)$ steps using Leung and Shende's algorithm [8]. Sorting in step (v) takes $O(n^{4/5})$ time. Step (vi) also can be finished in time $O(n^{4/5})$ because the actual destination of any key can be at the most one block away from where it is after step 4 (cf. Lemma 1). Thus, we have the following result:

Theorem 2 *Sorting on an $n \times n$ mesh with buses can be performed in $n + o(n)$ steps w.h.p.*

4. Open Problems

We have shown that the parallel model M_w can perform permutation routing as efficiently as the model M_b in the linear case. Our results for the 2D case

are weaker, and it is open if they can be improved. It is also open whether the randomized sorting result for M_b (Theorem 2) extends to the model M_w .

References

1. S. Cheung and F. C. M. Lau, A lower bound on permutation routing in meshes with buses, *Info. Proc. Lett.*, 1993.
2. C. Kaklamanis and D. Krizanc, Optimal Sorting on Mesh Connected Processor Arrays, in *Proc. 4th Annual ACM Symposium on Parallel Algorithms and Architectures*, San Diego, CA, 1992.
3. C. Kaklamanis, D. Krizanc, L. Narayanan and Th. Tsantilas, Randomized Sorting and Selection on Mesh-Connected Processor Arrays, in *Proc. 3rd Annual ACM Symp. on Parallel Algorithms and Architectures*, July 1991.
4. V. K. Prasanna Kumar, Communication complexity of various VLSI models, Ph. D. Thesis, Dept. of Computer Science, Pennsylvania State University, 1983.
5. M. Kunde, Concentrated Regular Data Streams on Grids: Sorting and Routing Near to the Bisection Bound, in *Proc. 32nd Annual IEEE Symposium on Foundations of Computer Science*, 1991, pp. 141-150.
6. F. T. Leighton, *Introduction to Parallel Algorithms and Architectures: Meshes, Trees, Hypercubes*, Morgan-Kaufmann Publishers, San Jose, CA, 1992.
7. J. Y-T. Leung and S. Shende, Packet Routing on Square Meshes with Row and Column Buses, in *Proc. Third IEEE Symposium on Parallel and Distributed Processing*, Dallas, Texas, December 1991, pp. 834-837.
8. J. Y-T. Leung and S. Shende, On Multi-Dimensional Packet Routing for Meshes with Buses, to appear in *J. Parallel and Distributed Computing*, 1993.
9. S. Rajasekaran, k - k Routing, k - k Sorting, and Cut Through Routing on the Mesh, Technical Report, Department of CIS, University of Pennsylvania, Philadelphia, PA 19104, October 1991. (Some results presented in *Proc. 4th Annual ACM Symp. on Parallel Algorithms and Architectures*, San Diego, CA 1992.)
10. S. Rajasekaran and T. McKendall, Randomized Routing and Sorting on the Reconfigurable Mesh, Technical Report MS-CIS-92-36, University of Pennsylvania, Philadelphia, PA 19104, May 1992.
11. S. Rajasekaran and Th. Tsantilas, Optimal Routing Algorithms for Mesh Connected Processor Arrays, *Algorithmica* 8, 1992, pp. 21-38.
12. R. Reischuk, Probabilistic Parallel Algorithms for Sorting and Selection, *SIAM Journal of Computing*, 14(2), 1985, pp. 396-411.
13. Q. F. Stout, Meshes with Multiple Buses, in *Proc. 27th IEEE Symp. on Foundations of Computer Science*, 1986, pp. 264-273.