Sorting on the OTIS-Mesh

Andre Osterloh[*]
Technical University of Ilmenau
Department for Automata and Formal Languages
PF 100565, D–98684 Ilmenau, Germany
osterloh@theoinf.tu-ilmenau.de

Abstract

In this paper we present sorting algorithms on the recently introduced N^2 processor OTIS-Mesh, a network with diameter $4\sqrt{N}-3$ consisting of N connected meshes of size $\sqrt{N}\times\sqrt{N}$. We show that k-k sorting can be done in $8\sqrt{N}+O(N^{\frac{1}{3}})$ steps for $k=1,2,3,4$ and in $2k\sqrt{N}+O(kN^{\frac{1}{3}})$ steps for $k>4$ with constant buffersize for all k.

We show how our algorithms can be modified to achieve $4\sqrt{N}+O(N^{\frac{1}{3}})$ steps for $k=1,2,3,4$ and $k\sqrt{N}+O(kN^{\frac{1}{3}})$ steps for $k>4$ in the average case. Finally we show a lower bound of $\max\{4\sqrt{N},\frac{1}{\sqrt{2}}k\sqrt{N}\}$ steps for k-k sorting.

1. Introduction

Several models for parallel machines were studied in the past and it has turned out that no model ideally fits for all applications. Especially well studied topologies is the mesh of processors or mesh-connected array which is a simple architecture that fulfills the demands of VLSI technology quite well. The two-dimensional mesh is ideally suited for several regular problems arising in the context of matrix calculations (multiplication etc.) and has also benefits for fundamental problems like sorting and routing.

However, there are several disadvantages of the two-dimensional mesh, e.g. its large diameter which gives a natural relatively large lower bound for just these basic problems like routing and sorting, or the small bisection width.

The OTIS-Mesh can be seen as an attempt to overcome the disadvantages of the mesh while trying to keep the benefits of the mesh.

The Optical Transpose Interconnection System (OTIS) proposed in [5] is a hybrid system using optical and electronical interconnections. The processors in the OTIS structure are partitioned in groups where the connections within the groups are realized by electronic links and the connections among the groups are realized by optical links.

Every group can be realized as mesh, torus, hypercube, etc. resulting in the OTIS-Mesh, OTIS-Torus, OTIS-Hypercube, etc. An $N\times N$ OTIS-Mesh consists of N meshes of size $\sqrt{N}\times\sqrt{N}$. The N meshes (groups) are connected such that if G represents a mesh and P represents a processor in mesh G, the processor (G,P) is connected with (P,G) a processor represented by G in the mesh represented by P (For an exact description see the next section.).

The architecture of the OTIS-Mesh could be either SIMD or MIMD. We will use the MIMD model where any processor can communicate with all of its neighbours in the same time step.

Several algorithms have been developed for the OTIS-Mesh. In [9] it is shown that an $N\times N$ OTIS-Mesh can simulate a four-dimensional mesh with constant delay. Sahni and Wang in [7] have given fast algorithms to route BPC-permutations on the OTIS-Mesh. In [8] several basic algorithms (broadcast, prefix sum, rank, sorting etc.) for the OTIS-Mesh are presented. Their 1-1 sorting algorithm is based on Leighton's column sort and needs $11\sqrt{N}+o(\sqrt{N})$ steps. They did not give the buffersize used by their algorithm. Rajasekaran and Sahni [6] have developed randomized algorithms for 1-1 routing, selection, and 1-1 sorting. Their algorithms run in time $4\sqrt{N}+o(\sqrt{N})$, $6\sqrt{N}+o(\sqrt{N})$, and $8\sqrt{N}+o(\sqrt{N})$, respectively.

In this paper we study sorting problems. We present an algorithm for k-k sorting, where each processor contains k packets initially and finally. Our algorithm needs $8\sqrt{N}+O(N^{\frac{1}{3}})$ steps for $k=1,2,3,4$ with buffersize $3,5,7,8$ and $2k\sqrt{N}+O(kN^{\frac{1}{3}})$ steps for $k>4$. For $k\geq 8$ the algorithm needs a buffersize of $k+4$ and for $k=5,6,7$ it needs a buffersize of at most 11. For $k=1$ we achieve the running time of the best randomized algorithm so far [6] while using a queue size of only 3. Our algorithm is not a derandomized version of the algorithm presented in [6]. For $k=1$ the running time of our algorithm is at most by

[*] This research was supported in part by the DFG-Project Ku 658/8-3

a factor 2 away from the optimal value, because the diameter of the OTIS-Mesh is $4\sqrt{N}-3$ [7]. For $k \geq 2$ the k-k sorting problem was not considered before for the OTIS-Mesh. The presented algorithm can be easily modified to solve the k-k sorting problem on the OTIS-Mesh in *half* of the running time in the average, i.e. $4\sqrt{N}+O(N^{\frac{1}{3}})$ steps for $k = 1,2,3,4$ and $k\sqrt{N}+O(kN^{\frac{1}{3}})$ steps for $k > 4$. Obviously our sorting algorithms solve also the k-k routing problem within the same number of steps.

We further show that every algorithm that solves the k-k sorting(routing) problem needs at least $\max\{4\sqrt{N}-3, \frac{1}{\sqrt{2}}k\sqrt{N}\}$ steps.

Our paper is organized as follows. In Section 2 we describe the structure of the OTIS-Mesh and present how a four-dimensional mesh can be simulated on the OTIS-Mesh. This simulation is used in all of our algorithms. In Section 3 we present a technique how to distribute data fast in the OTIS-Mesh. In Section 4 we use the data distribution to present our sorting algorithms, then we show a lower bound for k-k sorting and finally we describe how we can half the running time in the average case.

2. Preliminaries

For the rest of the paper we set $n := \sqrt{N} \in \mathbb{N}$ and choose $\varepsilon \in \mathbb{R}$ such that $0 < \varepsilon < 1$.

2.1. Structure of the OTIS-Mesh

An $N \times N$ OTIS-Mesh network consists of n^4 processors. The n^4 processors are grouped in N meshes of size $n \times n$. The connections between the N meshes are built according to the OTIS law: a processor P of group G is connected to a processor G of group P, $0 \leq G, P < N$. A processor index P of a processor p is given as a pair (p_x, p_y), where p_x is the row position and p_y is the column position of p within its mesh. A mesh index G is also given as a pair (g_x, g_y) where g_x and g_y are the row and column indices of the meshes if they are grouped as an $n \times n$ mesh. As a result of this interpretation each processor in the OTIS-Mesh can be labeled with a quadruple (i,j,k,l) with $0 \leq i,j,k,l < n$.

All connections of the network are bidirectional. The connections in the meshes are called *intrablock links* and connections between the meshes are called *interblock links* or *OTIS-links*. The intrablock links are given by the topology of the meshes, i.e. for $0 \leq i,j,k,k',l,l' < n$ processor (i,j,k,l) is connected with processor (i,j,k',l') iff $|k-k'|+|l-l'| = 1$. The OTIS-links are given as follows, for $0 \leq i,j,k,l < n$ processor (i,j,k,l) is connected with processor (k,l,i,j). It is easy to see that the diameter of the OTIS-Mesh is $4n-3$ [7]. In Figure 1 you can see a 4×4

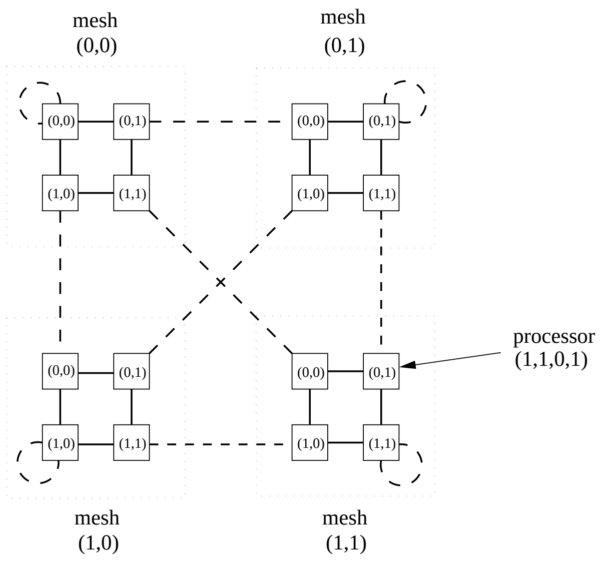

Figure 1. A 4×4 OTIS-Mesh.

OTIS-Mesh. The intrablock links are shown as solid lines and the OTIS-links are shown as dashed lines.

2.2. Simulation of a Four-dimensional Mesh

The simulation of a four-dimensional mesh with an OTIS-Mesh was first described in [9]. Because the simulation is important for our algorithms we repeat here the main ideas of it. In the simulation the processor (i,j,k,l) of the $n \times n \times n \times n$ four-dimensional mesh is identified with the processor (i,j,k,l) of the OTIS-Mesh, $0 \leq i,j,k,l < n$. The moves $(i,j,k \pm 1,l)$ and $(i,j,k,l \pm 1)$ of the four-dimensional mesh can be simulated by one move in the meshes of the OTIS-Mesh. The moves $(i \pm 1,j,k,l)$ and $(i,j \pm 1,k,l)$ are simulated by moves using the intrablock and OTIS-links. For example, the move $(i,j+1,k,l)$ can be simulated by: $(i,j,k,l) \longrightarrow (k,l,i,j) \longrightarrow (k,l,i,j+1) \longrightarrow (i,j+1,k,l)$. It is easy to see that this describes an embedding of the four-dimensional mesh in the OTIS-Mesh with congestion 4 and dilation 3. So the following holds:

Theorem 1 *Every algorithm that needs $f(n)$ steps and buffersize $b(n)$ on a $n \times n \times n \times n$ mesh can be performed in $O(f(n))$ steps and buffersize $\max\{2 * b(n)+1, b(n)+4\}$ on the OTIS-Mesh .*

Proof. Use the simulation technique described above. For all PUs in the four dimensional mesh number the neighbors uniformly from 1 to 8. To simulate the communication of neighboring PUs in the mesh every PU first communicate with neighbor 1 than with neighbor 2 etc., so every step of the mesh can be simulated within a constant number of steps in the OTIS-Mesh . If $b(n) < 4$ the worst case situation is

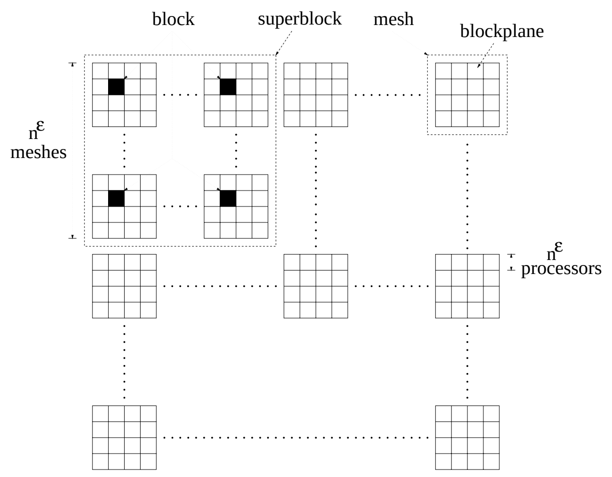

Figure 2. An n^4 processor OTIS-Mesh.

that a PU receive packets from their neighbors before it can sent packets and that a packet must be transported via this PU. So we get buffersize of at most $b(n) + b(n) + 1$. For the case $b(n) > 3$ notice that the congestion is 4 so the buffersize can be bounded by $b(n) + 4$. □

2.3. Substructures in the OTIS-Mesh

In this section we define *meshes*, *blocks*, *superblocks*, and *blockplanes*. A block is simply the realization of a four-dimensional area of processors in the OTIS-Mesh. A superblock is a collection of blocks and a blockplane is the cut of a block and a mesh. We now give the formal definitions.

For $0 \le a, b < n$ the mesh $M(a, b)$ consists of the processors (a, b, c, d) with $0 \le c, d < n$. For $0 \le a, b, c, d < n^{1-\varepsilon}$ the *block* $B(a, b, c, d)$ of size n^ε consists of the processors (i, j, k, l) with $an^\varepsilon \le i < (a+1)n^\varepsilon$, $bn^\varepsilon \le j < (b+1)n^\varepsilon$, $cn^\varepsilon \le k < (c+1)n^\varepsilon$, $dn^\varepsilon \le l < (d+1)n^\varepsilon$. For $0 \le a, b < n^{1-\varepsilon}$ the *superblock* $SB(a, b)$ consists of the blocks $B(a, b, c, d)$ with $0 \le c, d < n^{1-\varepsilon}$. If not empty, we call the cut of a block $B(a, b, c, d)$ and a mesh $M(i, j)$ *blockplane* and denote it by $BP(i, j, c, d)$. Notice that a cut of a block $B(a, b, c, d)$ and a mesh $M(i, j)$ is not empty iff $an^\varepsilon \le i < (a+1)n^\varepsilon$ and $bn^\varepsilon \le j < (b+1)n^\varepsilon$. In Figure 2 you can see an n^4 processor OTIS-Mesh with n^2 meshes of size $n \times n$. The n^2 meshes are grouped as an $n \times n$ mesh. In Figure 2 meshes, blocks, superblocks, and blockplanes are visualized. The filled black boxes in the superblock built one block. Note that the number of blockplanes in a block is $n^{2\varepsilon}$ and that the number of blocks in the OTIS-Mesh is $n^{4(1-\varepsilon)}$.

Now we introduce indexings for the meshes, blocks, superblocks, blockplanes, and processors. We will use row-major indexing for most of the structures: the processors within a blockplane, the blockplanes within a block, the meshes within a superblock, the blocks within a superblock, and the superblocks within the OTIS-Mesh. In Section 3 we use a blocked row-major indexing for the blocks, i.e. the index j of a block B within the OTIS-Mesh can be written as $j = j_1 n^{2(1-\varepsilon)} + j_2$ where j_1 is the index of the superblock SB in which B lies and j_2 is the index of B within that superblock. For example the index of the block $B(a, b, c, d)$ within the OTIS-Mesh is $(an^{1-\varepsilon} + b)n^{2(1-\varepsilon)} + cn^{1-\varepsilon} + d$. In Section 4 we use an arbitrary continuous indexing of the blocks. To handle indices we need the following definitions. For $a, b, c, x \in \mathbb{N}$ with $a = bx + c$ and $c < x$ we define $a \text{ div } x := b$ and $a \bmod x := c$. Finally we define for $x \in \mathbb{N}$ the set $[x] := \{i \in \mathbb{N} | i < x\}$.

3. Data Movement

In this section we describe how a kind of an all-to-all mapping [3] can be implemented on the OTIS-Mesh. An all-to-all mapping is a distribution of data within the network by sending pieces of data (so called bricks) from each block to each other block. For this purpose we give an algorithm that distribute the blockplanes of a block to all blocks. This is done for all blocks in parallel.

To achieve our target we first perform an 1-1 routing of the blockplanes within the meshes to distribute the blockplanes of a block in its superblock. After that we use the OTIS-links. This step distribute the packets of a blockplane to the blockplanes of one block such that every blockplane gets one packet. Then we perform an 1-1 routing in the blocks to collect this packets in one blockplane of the block. Finally we perform an 1-1 routing of the blockplanes within the meshes to distribute the blockplanes to the blocks in their destination superblock.

Algorithm Alg1: Blockplane distribution

1. Within all superblocks do: Transport the packets of blockplane j of block i to blockplane j of block $(i + (j \text{ div } n^\varepsilon)) \bmod n^{2(1-\varepsilon)}$, $0 \le j < n^{2\varepsilon}$, $0 \le i < n^{2(1-\varepsilon)}$.

2. Use the OTIS-links.

3. Within all blocks do: Transport the packets from processor j of blockplane k to processor k of the blockplane j, $0 \le j, k < n^{2\varepsilon}$

4. Within all superblocks do: Transport the packets of blockplane j of block i to blockplane j of block $(i + (j \bmod n^\varepsilon)) \bmod n^{2(1-\varepsilon)}$, $0 \le j < n^{2\varepsilon}$, $0 \le i < n^{2(1-\varepsilon)}$.

Theorem 2 *Algorithm* **Alg1** *can be performed in* $4n + O(n^{\varepsilon})$ *steps with queue size 3.*

Proof. Step 2 needs 1 step. Step 3 is an 1-1 routing within a block. An 1-1 routing in an $n^{\varepsilon} \times n^{\varepsilon} \times n^{\varepsilon} \times n^{\varepsilon}$ mesh can be performed in $O(n^{\varepsilon})$ steps with no additional buffer [2]. Using Theorem 1 Step 3 can be performed in $O(n^{\varepsilon})$ steps with queue size 3. Step 1 and Step 4 are 1-1 routings in the mesh. 1-1 routing on a mesh can be performed in $2n + O(1)$ steps with queue size 2 [1]. □

Now we consider the case that every PU holds k packets.

Theorem 3 *If every PU holds* k *packets then* **Alg1** *can be performed in* $4n + O(n^{n^{\varepsilon}})$ *steps for* $k = 1, 2, 3, 4$ *with queue size* $3, 5, 7, 8$, *and in* $2kn + O(kn^{n^{\varepsilon}})$ *steps for* $k > 4$ *with queue size* $k + 4$ *for* $k \geq 8$ *and queue size of at most* 11 *for* $k = 5, 6, 7$.

Proof. First we consider the case $k \geq 8$. Step 1 and Step 4 are k-k routings and can be performed in $\frac{kn}{2} + O(kn^{n^{\varepsilon}})$ steps with queue size k[1]. Step 2 needs k steps. Using the result of Theorem 1 Step 3 can be performed in $O(kn^{n^{\varepsilon}})$ steps with queue size $k + 4$.

Now we consider the case $k < 8$: A closer look at Step 1 and Step 4 shows that the blockplanes are just shifted by h positions with respect to a row-major indexing of the blockplanes within the meshes ($0 \leq h < n^{2(1-\varepsilon)}$). This routing can be done by first performing permutations in the rows and then performing permutations in the columns of the mesh. Using odd-even transposition sort to perform the permutation routing in the rows(columns) and the technique of overlapping Step 1 and Step 4 can be done in $2n + O(1)$ steps each, with queue size $2k$ for $k = 2, 3, 4$, and in $2kn + O(1)$ steps with queue size of at most 10 for $k = 5, 6, 7$. The rest follows from Theorem 1. □

In the following we observe where the packets of blockplanes are moved to by algorithm **Alg1**. We use a blocked row-major indexing of the blocks.

Lemma 4 **Alg1** *transport the packets from blockplane* j *of block* i *to blockplane* j *of block* $((i + (j \text{ div } n^{\varepsilon})) \bmod n^{2(1-\varepsilon)})n^{2(1-\varepsilon)} + ((i \text{ div } n^{2(1-\varepsilon)}) + (j \bmod n^{\varepsilon})) \bmod n^{2(1-\varepsilon)}$, $0 \leq j < n^{2\varepsilon}$, $0 \leq i < n^{4(1-\varepsilon)}$.

Proof.(Sketch) Observe step by step where the packets of blockplane j of block i are moved to by algorithm **Alg1**. □

Now we consider the case $\varepsilon = \frac{2}{3}$. For this case the number of blocks in the OTIS-Mesh is equal to the number of blockplanes of a block.

Theorem 5 *For* $\varepsilon = \frac{2}{3}$ **Alg1** *realizes an all-to-all mapping, in the sense that* **Alg1** *distribute the packets of all blockplanes of one block so that every block gets exactly the packets of one blockplane. This is done for all blocks in parallel.*

Proof. For all $0 \leq i < n^{\frac{4}{3}}$, $j \mapsto ((i + (j \text{ div } n^{\frac{2}{3}})) \bmod n^{\frac{2}{3}})n^{\frac{2}{3}} + ((i \text{ div } n^{\frac{2}{3}}) + (j \bmod n^{\frac{2}{3}})) \bmod n^{\frac{2}{3}}$ is a permutation of the set $[n^{\frac{4}{3}}]$. □

For block $B(a,b,c,d)$ let ${}^{g}transport_{B(a,b,c,d)}(j)$ be the index of the block to which the packets of blockplane j of block $B(a,b,c,d)$ are transported to by **Alg1** with respect to the blocked row-major indexing g of the blocks in the OTIS-Mesh. For all blocks $B(a,b,c,d)$ the function ${}^{g}transport_{B(a,b,c,d)}$ is a permutation of the set $[n^{\frac{4}{3}}]$. Now consider an arbitrary indexing f of the blocks then the following holds ${}^{f}transport_{B(a,b,c,d)}(j)$ $= f(g^{-1}({}^{g}transport_{B(a,b,c,d)}(j)))$.

4. Sorting

Our algorithm sorts the OTIS-Mesh with respect to a continuous indexing f of the blocks in the interpretation of the OTIS-Mesh as a four-dimensional mesh. The indexing h within the blocks can be chosen arbitrarily.

The algorithm **Alg2** is based on the principle of sorting with all-to-all mappings [3]. A raw description of sorting with all-to-all mappings looks as follows. Divide the network into blocks, number them with a continuous indexing, sort the blocks, perform an all-to-all mapping, sort the blocks, perform an all-to-all mapping and finally sort neighbouring blocks. The time complexity of such a scheme is dominated by the complexity of the all-to-all mapping.

This scheme can be applied to the OTIS-Mesh. For the OTIS-Mesh we use blocks of size $n^{\frac{2}{3}}$ and sort them with the simulation technique of Section 2.2. As all-to-all mapping we use **Alg1** for $\varepsilon = \frac{2}{3}$.

Algorithm Alg2: Sort OTIS-Mesh.

1. (a) Sort all blocks of size $n^{\frac{2}{3}}$ with respect to h.
   (b) Within all blocks $B(a,b,c,d)$ of size $n^{\frac{2}{3}}$ do: Transport the packets with index i with respect to h to blockplane ${}^{f}transport^{-1}_{B(a,b,c,d)}(i \bmod n^{\frac{4}{3}})$, $0 \leq a,b,c,d < n^{\frac{1}{3}}$, $0 \leq i < n^{\frac{8}{3}}$.
2. Perform an all-to-all mapping (**Alg1**, $\varepsilon = n^{\frac{2}{3}}$).
3. (a) Sort all blocks of size $n^{\frac{2}{3}}$ with respect to h.
   (b) Within all blocks $B(a,b,c,d)$ of size $n^{\frac{2}{3}}$ do: Transport the packets with index i with respect to h to blockplane ${}^{f}transport^{-1}_{B(a,b,c,d)}(i \text{ div } n^{\frac{4}{3}})$, $0 \leq a,b,c,d < n^{\frac{1}{3}}$, $0 \leq i < n^{\frac{8}{3}}$.
4. Perform an all-to-all mapping (**Alg1**, $\varepsilon = n^{\frac{2}{3}}$).
5. Sort all pairs of blocks $(2i, 2i+1)$, $0 \leq i < \frac{n^{\frac{4}{3}}}{2}$. Sort all pairs of blocks $(2i-1, 2i)$, $0 < i < \frac{n^{\frac{4}{3}}}{2}$, (pairs of blocks with respect to f).

Theorem 6 *For $k = 1, 2, 3, 4$* **Alg2** *solve the k-k sorting problem on the OTIS-Mesh in $8n + O(n^{\frac{2}{3}})$ steps with queue size $3, 5, 7, 8$. For $k = 5, 6, 7$* **Alg2** *solve the k-k sorting problem on the OTIS-Mesh in $2kn + O(n^{\frac{2}{3}})$ steps with maximal queue size 11 and for $k \geq 8$* **Alg2** *solve the k-k sorting problem on the OTIS-Mesh in $2kn + O(n^{\frac{2}{3}})$ steps with queue size $k + 4$.*

Proof. That **Alg2** solve the k-k sorting problem follows from [3]. Step 1a and Step 3a are k-k sortings of blocks and Steps 1b and Step 3b are k-k routings within the blocks and could be performed with the simulation technique of Section 2.2. So Steps 1, 3, 5 can be achieved in $O(n^{\frac{2}{3}})$ steps with the queue size given in the Theorem. For Steps 2, 4 see Theorems 2,3. □

4.1. Lower bound

Now we proof a lower bound for k-k routing and k-k sorting. The proof is based on a bisection argument.

Theorem 7 *Every algorithm that solve the k-k sorting (routing) problem on the OTIS-Mesh need at least $\max\{4n - 3, \frac{1}{\sqrt{2}} \approx 0.707\ kn\}$ steps.*

Proof. Look at Figure 3. There are two areas of processors denoted with X and Y. For $c = \frac{1}{\sqrt{2}}$ both areas consist of c^2n^4 processors. Note that there are no OTIS-links between processors of area X and processors of area Y. So a packet that wants to travel from a processor in area X to a processor in area Y has to cross one of the dashed lines (called border). There are cn^3 intrablock links crossing the border. Hence to transport kc^2n^4 packets from area X to area Y at least $\frac{kc^2n^4}{cn^3}$ steps are needed. For $c = \frac{1}{\sqrt{2}}$ we get the desired result. Note that $4n - 3$ is the diameter of the OTIS-Mesh. □

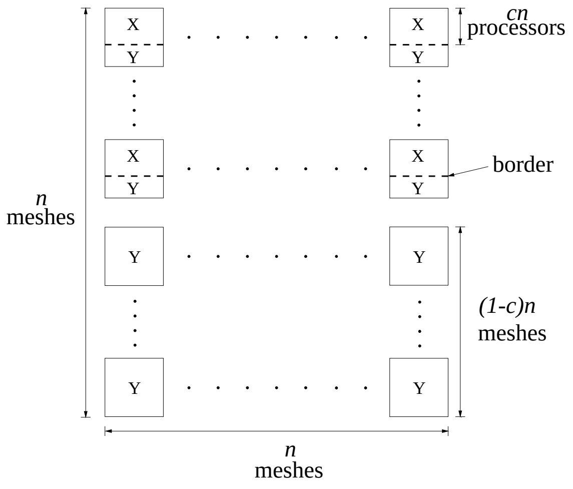

Figure 3. An OTIS-Mesh divided in two areas X and Y.

4.2. Average Case Sorting

In [4] a way is presented to get algorithms that sort and route on mesh-based computers fast on average. Their algorithms use a sorting scheme based on all-to-all mappings. The main idea is based on the following observation. Assume for a moment that the input for the sorting problem consists only of zeros and ones. Then the first all-to-all mapping distribute the input uniformly over the blocks. So you have nearly the same number of ones and zeros in every block. Now assume that all possible loads of zeros and ones are uniformly distributed. If you pick one load arbitrarily then the number of zeros and ones in the blocks differ only by a small amount with high probability. As a consequence an algorithm that sort blocks, perform an all-to-all mapping, and finally sort neighbouring blocks solve the sorting problem with high probability. Now let A be an algorithm that solve the sorting problem on the considered network. To get an algorithm that sort fast on average you must simply perform one more step. You have to check whether all packets are in order, and if not you have to sort with algorithm A. If A is not to slow the average case time complexity of the whole algorithm is dominated by the first three steps.

Let b be the number of packets in one block and k be the number of blocks. To make sure that the first three steps solve the sorting problem with high probability $\frac{b}{k^2} \in \Omega(n^{\alpha})$ for an $\alpha > 0$ must hold (see Lemma 2, [4]). For blocks of size $n^{\frac{2}{3}}$ we get $\frac{b}{k^2} \in O(1)$ but for blocks of size $n^{\frac{3}{4}}$ we have $\frac{b}{k^2} \in \Omega(n)$. For $\varepsilon = \frac{3}{4}$ **Alg1** sends packets from $n^{\frac{1}{2}}$ blockplanes of a block to all blocks. In this case every blockplane consists of $n^{\frac{3}{2}}$ processors, hence every block sends n^2 packets to every block. This is exactly one brick (bricksize= $\frac{b}{k}$ see [4]).

For the algorithm A in the description above modify **Alg2** such that it uses blocks of size $n^{\frac{3}{4}}$.

Theorem 8 *On the OTIS-Mesh k-k sorting can be performed in $4n + o(n)$ steps for $k = 1, 2, 3, 4$ and in $kn + o(n)$ steps for $k \geq 5$ on average.* □

5. Conclusion

In this paper a fast deterministic algorithm for sorting on the OTIS-Mesh is presented. An unsolved problem is to find deterministic algorithms for k-k sorting matching the known lower bounds.

References

[1] M. D. Grammatikakis, D. F. Hsu, and J. F. Sibeyn. Packet routing in fixed-connection networks: A survey. *Journal of Parallel and Distributed Computing*, 54(2):77–132, 1 Nov. 1998.

[2] M. Kunde. Routing and sorting on mesh-connected arrays. In J. H. Reif, editor, *Proceedings of the 3rd Aegean Workshop on VLSI Algorithms and Architectures*, pages 423–433. LNCS 319. Springer, July 1988.

[3] M. Kunde. Block gossiping on grids and tori: Deterministic sorting and routing match the bisection bound. In T. Lengauer, editor, *First Annual European Symposium (ESA'93)*, pages 272–283. LNCS 726. Springer, Sept. 30–Oct. 2, 1993.

[4] M. Kunde, R. Niedermeier, K. Reinhardt, and P. Rossmanith. Optimal average case sorting on arrays. In E. W. Mayr and C. Puech, editors, *Proceedings of the 12th Annual Symposium on Theoretical Aspects of Computer Science (STACS'95)*, pages 503–514. LNCS 900. Springer-Verlag, Mar. 2–4 1995.

[5] G. C. Marsden, P. J. Marchand, P. Harvey, and S. C. Esener. Optical transpose interconnection system architectures. *Optics Letters*, 18(13):1083–1085, 1993.

[6] S. Rajasekaran and S. Sahni. Randomized routing, selection, and sorting on the OTIS-mesh. *IEEETPDS: IEEE Transactions on Parallel and Distributed Systems*, 9(9):833–840, 1998.

[7] C. Wang and S. Sahni. BPC permutations on the OTIS-mesh optoelectronic computer. In *Proceedings on the Fourth International Conference on Massively Parallel Processing Using Optical Interconnections (MPPOI'97)*, pages 130–135, 1997.

[8] C. Wang and S. Sahni. Basic operations on the OTIS-mesh optoelectronic computer. *IEEETPDS: IEEE Transactions on Parallel and Distributed Systems*, 9(12), 1998.

[9] F. Zane, P. J. Marchand, R. Paturi, and S. C. Esener. Scalable network architectures using the optical transpose interconnection systems (OTIS). In *Proceedings on the Fourth International Conference on Massively Parallel Processing Using Optical Interconnections (MPPOI'96)*, pages 114–121, 1996.