Towards Practical Permutation Routing on Meshes

Michael Kaufmann* Uli Meyer† Jop F. Sibeyn‡

Abstract

We consider the permutation routing problem on two-dimensional $n \times n$ meshes. To be practical, a routing algorithm is required to ensure very small queue sizes Q, and very low running time T, not only asymptotically but particularly also for the practically important n up to 1000. With a technique inspired by a scheme of Kaklamanis/Krizanc/Rao, we obtain a near-optimal result: $T = 2 \cdot n + \mathcal{O}(1)$ with $Q = 2$. Although Q is very attractive now, the lower order terms in T make this algorithm highly impractical. Therefore we present simple schemes which are asymptotically slower, but have T around $3 \cdot n$ for *all* n and Q between 2 and 8.

1 Introduction

Communication between processing units (**PUs**) in a network is performed by exchanging packets of information. Since the network is sparse, due to physical constraints on the number and length of links, the packets have to travel through intermediate nodes. **Packet routing** is concerned with the organization of the movement of the packets in a network. The efficiency of a packet-routing protocol is measured by (1) the time that passes until all the routing requests are completed; and (2) the size of auxiliary memory in each PU. The time is measured by the maximum number T of routing steps, and the memory by the maximum number Q of packets that may be queued simultaneously in a PU. As we think of PUs as small nodes with limited storage capacity, it is very important to have algorithms that work with small Q. Furthermore, the larger the queues are, the longer it takes to insert and extract packets from them: the assumption that managing the queues takes no time is realistic only if the queues are very small. Another criterion for a practical algorithm is simplicity: in applications an algorithm with a few instructions always outperforms a complicated scheme which appears better in theory.

*Fakultät Informatik, Universität Tübingen, Sand 13, 72076 Tübingen, Germany. E-mail: mk@informatik.uni-tuebingen.de

†Max-Planck-Institut für Informatik, Im Stadtwald, 66123 Saarbrücken, Germany. E-mail: umeyer@mpi-sb.mpg.de

‡Max-Planck-Institut für Informatik, Im Stadtwald, 66123 Saarbrücken, Germany. E-mail: jopsi@mpi-sb.mpg.de Partially supported by EC Cooperative Action IC-1000 (Project ALTEC: Algorithms for Future Technologies).

We consider routing permutations on a two-dimensional $n \times n$ MIMD mesh. In a **permutation routing problem**, every PU is source and destination of precisely one packet. A routing algorithm is called **optimal** if $T = 2 \cdot n - 2$, the diameter of the mesh, and **near-optimal** if $T = 2 \cdot n + \mathcal{O}(1)$. Recently, routing on meshes has attracted a considerable amount of attention. The first routing algorithms which required close to $2 \cdot n$ steps were given by Kunde [5] and Rajasekaran and Tsantilas [10]. Leighton, Makedon and Tollis [6] presented the first deterministic algorithm with optimal routing time and constant size queues. This paper is of a great theoretical importance but the maximum queue size is impractically large ($Q = 1008$ according to [9]). Rajasekaran and Overholt [9] reduced Q to about 150 (for comments see [1]). Further improvements are given in [1] and [12]: we presented routing algorithms with $T = 2 \cdot n - 2$ and $Q = 33$, and with $T = 2 \cdot n + \mathcal{O}(1)$ and $Q = 12$.

In this paper we take a different approach. It appears that the line of algorithms from [6] over [9] to [1] and [12] has come to an end. We believe that no further substantial improvements can be achieved by developing the routing scheme, the scattering technique, or the spreading technique. Therefore, we step back somewhat and consider a combination of the algorithms of Rajasekaran/Tsantilas [10], Kaklamanis/Krizanc/Rao [2] and Kaufmann/Sibeyn/Suel [4]. This gives an algorithm with really short queues, $Q = 2$ and $T = 2 \cdot n + \mathcal{O}(1)$. However, the additional constant is impracticably large. Therefore we consider schemes which are simple and behave much better than any known algorithm for practically important sizes of the mesh: n between 16 and 1024. Variants give a trade-off between Q and T. These algorithms have been simulated to test their implementability, and to determine the actual number of routing steps.

In the remainder of the paper we first give an overview of basic ideas underlying the algorithms. Section 3 offers a basic algorithm with $T = 3 \cdot n + 41 \cdot n^{1/2}$ and $Q = 2$. Then this algorithm is refined to one with $T = 2 \cdot n + \mathcal{O}(n^{3/4})$ and $Q = 2$. In Section 4.5 we introduce critical and non-critical packets to obtain $T = 2 \cdot n + \mathcal{O}(1)$ and $Q = 2$. Hereafter we give the more practical schemes and present results of simulations thereof.

2 Preliminaries

Model Assumptions. As computer model we assume a two-dimensional $n \times n$ MIMD mesh without wrap-around connections. We refer to this machine simply by **mesh**. It consists of n^2 PUs, each of which is connected to (at most) four other PUs by a regular square grid. The PU at position (i, j) is referred to by $P_{i,j}$, where $P_{0,0}$ is in the upper-left corner. One **packet** of bounded length can be routed in each direction over one link during a **step**. Thus a PU may send and receive during a step (at most) four packets. Packets are never divided or combined. Packets carry information that enables the PUs to route them to their destination. It may happen that a packet has to wait a number of steps in some PU P. In the meantime other packets may enter P, and thus P may have to store packets in a queue of some size Q. In all queue-size analysis we assume that in addition to a queue, a PU has buffers connected to its in- and outgoing connections:

Assumption 1 *Packets that are passing by are not counted when determining Q. It is even possible to exchange a packet in a buffer and a packet in a queue.*

In the algorithms with $T = 2 \cdot n + o(n)$ and $Q = 2$ we assume that packets are 'consumed' upon arrival:

Assumption 2 *Packets that have reached there destination are not counted when determining Q.*

We want to stress that both assumptions appear natural and that they are implicitly made in the analysis of many other routing algorithms as well.

Scattering. Fast and efficient **scattering** subroutines, which redistribute the packets in $s \times s$ submeshes of an $n \times n$ mesh such that the number of packets from one row of the mesh with destination in one column of the mesh is minimized, are essential for the routing algorithms of this paper. The scattering subroutine must be **uni-axial** in order to fit into the routing algorithm: in any given step the routing must be performed either only over vertical links, or only over horizontal links. In [11] it is shown that

Lemma 1 *Uni-axial row-major (column-major) 1-1 sorting on $n \times n$ meshes with $Q = 2$ can be performed in $5^1\!/_2 \cdot n$ steps. 2-2 sorting with $Q = 2$ can be performed in $7^1\!/_4 \cdot n$ steps.*

One-Dimensional Routing. Our algorithms are composed of routing operations within one-dimensional subarrays. If during such a routing several packets are competing for the use of the same connection, the packet that has to go farthest in this direction gets priority: that is, we apply the **farthest-first strategy**.

In order to analyze the routing within the rows or columns we need the **routing lemma**, which gives the exact number of routing steps needed:

Lemma 2 [3] *Define for a given distribution of packets over the PUs $h_r(i, j) = \#\{$ packets passing from left to right through i and j $\}$, and let T_r be the number of required rightward routing steps. Then*

$$T_r = \max_{i<j}\{j - i + h_r(i, j) - 1\}.$$

Types of Packets and Subdivisions. Packets that need to move between opposite corner submeshes of size $a \times a$ are called **critical**, the other packets are called **non-critical**. In some of our algorithms, the non-critical packets perform a deterministic three-phase algorithm ('randomization' along the columns, and then greedily towards their destinations); the critical packets are routed recursively in some designated $a \times a$ area. Eventually a becomes too small to apply recursion. Then we finish with a sub-optimal algorithm assuring short queues, e.g., sorting with the algorithm of [11]. For reducing the queue sizes, we apply basic spreading techniques. Furthermore, the mesh is divided into $s \times s$ submeshes $S_{i,j}$, which are indexed as the PUs, starting with $S_{0,0}$ in the upper-left corner. Throughout this paper $m = n/s$. Let **row-bundle** i, $R_i = \cup_{j=0}^{m-1} S_{i,j}$; and let **column-bundle** j, $C_j = \cup_{i=0}^{m-1} S_{i,j}$. See Figure 1 for an illustration.

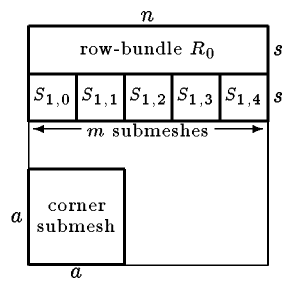

Figure 1: Subdivisions for the case $s = n/5$, $m = 5$.

3 Basic Algorithm

The algorithms which we present in the following sections can be viewed as deterministic versions of an improvement of the algorithm of Rajasekaran and Tsantilas [10]. Like all other recent permutation routing algorithms [5, 6, 9, 1], it consists of the greedy algorithm preceded with steps to reduce the arising queue sizes:

1. Route all packets along the columns to pseudo-random destinations within their columns.

2. Route all packets along the rows to their destination columns.

3. Route all packets along the columns to their destinations.

The essential point is how Phase 1 is worked out.

In a simple deterministic algorithm with $Q = 2$, the main phases are interleaved with local scattering phases:

Algorithm BASIC-ROUTE

1. In every submesh $S_{i,j}$, the packets are sorted on the indices of their destination column-bundles.

2. The packets are unshuffled along the columns: in every submesh $S_{i,j}$, $0 \leq i, j < m$, send the packet with rank r, standing in position (k, l), $0 \leq k, l < s$, to position (k, l) of $S_{(i+r) \bmod m, j}$.

3. In every submesh, the packets are sorted in column-major order on the indices of their destination column-bundles.

4. The packets are routed along the rows to the first PUs in their destination column-bundles holding less than two packets.

5. In every submesh, the packets are sorted in row-major order on the indices of their destination row-bundles.

6. The packets are routed along the columns to the first PUs in their destination submeshes holding less than two packets.

7. In every submesh, the packets are routed to their destinations.

In Step 2 any PU is destination of precisely one packet, if in Step 1 the sorting is performed with respect to the same indexing in all submeshes. The correctness follows from the following lemma and its analogue for Step 6:

Lemma 3 *If $s \geq (1 + \sqrt{5})/2 \cdot n^{1/2}$, then Step 4 can be performed as specified.*

Proof: Consider the packets with destination in some column-bundle C_j. Let $\alpha_{k,l}$ be the number of these packets, which initially reside in $S_{k,l}$. $\sum_{k,l} \alpha_{k,l} = s \cdot n$. At most $\lceil \alpha_{k,l}/m \rceil$ of the packets in $S_{k,l}$ are routed from submesh $S_{k,l}$ to $S_{i,l}$ in row-bundle R_i. After Step 2, there are in total at most $\sum_k \lceil \alpha_{k,l}/m \rceil$ packets in $S_{i,l}$ with destination in C_j. After Step 3 at most $\lceil (\sum_k \lceil \alpha_{k,l}/m \rceil)/s \rceil$ of them stand in the same row. In total there are at most $\sum_l \lceil (\sum_k \lceil \alpha_{k,l}/m \rceil)/s \rceil < m + m^2/s + s$ packets in a row with destination in C_j. So, in order to be able to perform Step 4, s must be taken so large that $m + m^2/s \leq s$. □

Theorem 1 *With $s \geq (1+\sqrt{5})/2 \cdot n^{1/2}$,* BASIC-ROUTE *performs permutation routing on an $n \times n$ mesh in $3 \cdot n + 25 \cdot s < 3 \cdot n + 41 \cdot n^{1/2}$ steps and with maximal queue size 2.*

Proof: Step 1 and Step 3 are 1-1 sortings and take $5^1\!/\!_2 \cdot s$ steps. In Step 2 packets travel at most a distance $n - s$. Step 5 is a 2-2 sorting and takes $7^1\!/\!_4 \cdot s$ steps. Step 7 can be performed by combining a 2-2 sorting with a trivial routing operation, which can be performed in $s/2$ steps: a packet which has to go to a PU with row-major index i, $0 < is^2 - 1$, within its submesh is given key $i \bmod (s^2/2)$. Then after sorting in row-major order, a packet stands either in its destination PU or $s/2$ rows above it. Summing we find $T = 3 \cdot n + 25 \cdot s$. □

There are some restrictions on n and s: s must be a power of two, and a divisor of n. For example, for $n = 384 = 12 \cdot 32$, we can take $s = 32$, and the routing takes $3 \cdot n + 25 \cdot n/12 \simeq 5^1\!/\!_{12} \cdot n$ steps. This appears bad, but so far no other routing algorithm achieves $Q = 2$ with less routing steps (for smaller n it is better to sort in $5^1\!/\!_2 \cdot n$ steps).

4 New Approach

We refine BASIC-ROUTE and obtain the first permutation routing algorithm with near-optimal performance and maximal queue size two. There are two main ideas: For the pseudo-randomization in Step 2 we choose a destination such that the sum of the distances that a packet has to travel during Step 2 and Step 6 does not exceed n. The second idea, going back on [10], is to coalesce the phases. In our approach this is particularly difficult since the routing phases are separated by the scattering phases (Step 3 and Step 5). If the routing phases are coalesced then the scattering cannot be performed as before. We show how to scatter disjointly with the same quality as a single scattering on all packets at the same time.

4.1 Modifying Phase 1

We start with a randomized version of the modified Phase 1. As in [2] the packets are randomized in **randomization ranges** that depend on both the source and the destination rows. But in [2] only the case of static randomization range is analyzed, although the concept of flexible ranges is mentioned. It has the great advantage that now the expected number of packets that are routed in Phase 1 to any row is strictly less than $2 \cdot n$.

A packet p in row i with destination in row j has randomization range

$$\mathcal{A}_{i,j} = \left[\max\{0, \frac{i+j-n}{2}\}, \min\{n, \frac{i+j+n}{2}\} \right].$$

Define $a_{i,j}$ to be the size of the range:

$$a_{i,j} = \min\{n, \frac{i+j+n}{2}\} \; - \; \max\{0, \frac{i+j-n}{2}\} \; + \; 1.$$

Every packet p is routed with probability $1/a_{i,j}$ to any row in its randomization range. The ranges are the largest subsets for which the sum of the lengths of the pathes in Phase 1 and in Phase 3 is at most n. Two typical examples are given in Figure 2.

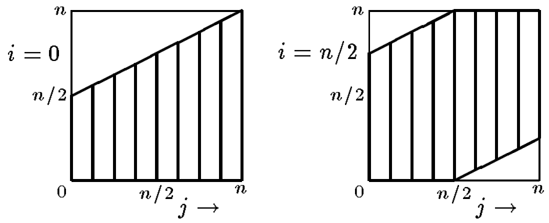

Figure 2: Randomization ranges as a function of j: left for $i = 0$, right for $i = n/2$.

Routing Time of Phase 2. By the **density** of the packets in a subset $\mathcal{S}$ of the PUs we mean the expected number of packets residing in $\mathcal{S}$ divided by $\#\mathcal{S}$. The density in a row is important because the routing lemma (Lemma 2) implies that, under condition that packets nor destinations lie concentrated, any distribution with density at most 2 can be routed in n steps. In the following we give a detailed analysis.

Consider the routing in some row i. Let x, y be an arbitrary pair of indices with $x < y$. According to Lemma 2, we can concentrate on the packets that have to travel from a column $j_1 \leq x$, to some column $j_2 \geq y$. Let $\mathcal{X}$ be the set of PUs in the leftmost x columns, and $\mathcal{Y}$ of the PUs in the rightmost $n-y$ columns. The Worst cases are obtained when $n \cdot \min\{x+1, n-y\}$ packets are given sources in $\mathcal{X}$ and destinations in $\mathcal{Y}$ such that the expected value of $h_r(x, y)$ is maximized. It can be shown that if $x > n - y - 1$, that then for $x' = x - 1$, $h_r(x, y) < h_r(x', y) + 1$: going from x' to x gives more freedom in the placement of the packets within $\mathcal{X}$, but the effect on the expected value of $h_r(x, y)$ is small. An analogous relation holds when $x < n-y-1$. Hence, we may concentrate on the cases $x = n - y - 1$. As the source and destination column of a packet p have no influence on the probability that p is randomized to row i, we may assume without loss of generality that the packets in column j have destination in column $n - j - 1$.

Hereby the problem is reduced to the one-dimensional problem of finding the permutation of $\{0, 1, \ldots, n-1\}$ that gives the maximal density in some i. Trying some examples, one will discover soon that this maximum is assumed for $i = n/2$ under the identity *Id*. We prove that this observation is correct. For a permutation π, let $dens_\pi(i)$ be the resulting density in a PU i under π. Let $dens(\pi) = \max_i\{dens_\pi(i)\}$.

Lemma 4 *$dens(\pi) \leq dens(Id)$, for all permutations $\pi : [0, n-1] \rightarrow [0, n-1]$.*

Proof: Consider some PU i. Suppose that $\pi(i) = j \neq i$, that $\pi(k) = i$, and that $i < j, k$. If $\pi \neq Id$, then such an i exists: i can be taken equal to the minimal value of a cycle of π. Let π' equal π except for $\pi'(i) = i$, and $\pi(k) = j$. If we show that $dens(\pi') \geq dens(\pi)$ then we are done, because then π can be transformed step-by-step to *Id*. Figure 3 gives a schematical representation

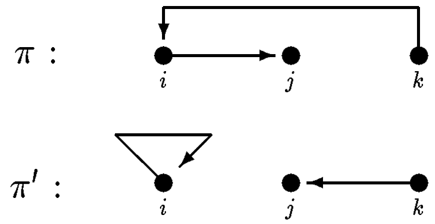

Figure 3: Situation under π and π'.

of the situation under π and π' (modulo the order of j and k). We must show that

$$\frac{1}{a_{i,j}} + \frac{1}{a_{k,i}} \leq \frac{1}{a_{i,i}} + \frac{1}{a_{k,j}}. \tag{1}$$

The proof would be easy if there would not appear maximas and minimas in the definition of $a_{i,j}$. Now several cases must be distinguished. Suppose that $j + k \leq n$. Let $a = 2 \cdot i + n + 2$, $b = j - i$, and $c = k - i$. Then (1) takes the form

$$\frac{2}{a+b} + \frac{2}{a+c} \leq \frac{2}{a} + \frac{2}{a+b+c}.$$

And, this is true, because $a \cdot (a+b+c) < (a+b) \cdot (b+c)$.

For the other cases, like $j+k > n$ and $i+j, i+k \leq n$, the proof can be given analogously. □

Lemma 5 *For the identity permutation on a one-dimensional array, sending the packets to a uniformly chosen PU from their randomization range, the density in any PU is less than $2 \cdot \ln 2 \simeq 1.386$.*

Proof: We analyze the densities in some PU k. For *Id* the expressions for the randomization ranges and their sizes become very simple: $\mathcal{A}_i = [0, i + n/2]$, $a_i = i + n/2 + 1$, for $0 \leq i < n/2$. $\mathcal{A}_i = [i - n/2 - 1, n-1]$, $a_i = 3/2 \cdot n - i + 1$, for $n/2 \leq i < n$. Only PU $n/2 - 1$ and PU $n/2$ lie in the ranges for all i. So, clearly $dens(k) \leq dens(n/2) < 2 \cdot \sum_{i=n/2}^{n-1} 1/(3/2 \cdot n - i) < 2 \cdot \int_{n/2}^{n} 1/(3/2 \cdot n - i) \mathrm{d}i = 2 \cdot \ln 2$. □

We conclude that sending the packets to a uniformly selected PU from their randomization ranges, gives in every row a distribution which can easily be routed in n steps.

Processor Densities. In a non-coalesced version of the algorithm, the maximal queue size is determined by the maximal densities that may arise in the PUs. We show that they are less than two, which is sufficient for proving that the maximal occurring queue size is two. The analysis for the real algorithm, in which the phases *are* coalesced, is given in Section 4.4.

Lemma 6 *For all permutations, the density in any PU is less than* 1.63.

Proof: We analyze the density in PU P_i at position (i, j) in some column j, $0 \leq j < n$. We consider for all PUs P_k, $0 \leq k < n$, in column j, the size $\alpha_{i,k}$ of the smallest randomization-range that includes P_i. This size is determined by the destination row of the packet starting in P_k. The maximum density in P_i satisfies $dens(P_i) \leq \sum_k 1/\alpha_{i,k}$.

Consider for given k, a packet p starting in P_k, with destination in some row l. The randomization range of p is maximal for $l = n - k$: $a_{k,n-k} = n$. $a_{k,l}$ decreases for l larger or smaller. Thus, $\alpha_{i,k}$ is assumed for the largest or for the smallest possible l, such that $i \in \mathcal{A}_{k,l}$. Without loss of generality we assume that $i < n/2$. Then the smallest l equals 0, for all k. The largest l equals $\min\{n-1, n+2\cdot i-k\}$. We have $a_{k,0} = (n+k)/2$, and $a_{k,\min\{n-1,n+2\cdot i-k\}} = \max\{n-i, n-k/2\}$. Hence, $\alpha_{i,k} = \min\{(n+k)/2, \max\{n-i, n-k/2\}\}$. This gives four possible ranges:

$$\begin{aligned}
\alpha_{i,k} &= n - i, \text{ if } k \geq \max\{2\cdot i, n - 2\cdot i\} \\
\alpha_{i,k} &= n - k/2, \text{ if } n/2 \leq k \leq 2\cdot i \\
\alpha_{i,k} &= (n+k)/2, \text{ if } 2\cdot i \leq k \leq n - 2\cdot i \\
\alpha_{i,k} &= (n+k)/2, \text{ if } k \leq \min\{2\cdot i, n/2\}.
\end{aligned}$$

The relations become particularly easy for $i = n/2$. For smaller i, some of the $\alpha_{i,k} = n - k/2$, are replaced by (larger) $\alpha_{i,k} = n - i$. Formalizing this argument, it follows that the maximal density occurs in $P_{n/2}$. We have $\alpha_{n/2,k} = (n+k)/2$, for $k \leq n/2$, and $\alpha_{n/2,k} = n - k/2$, for $k \geq n/2$. This gives $dens(P_{n/2}) = 2 \cdot \sum_{k=0}^{n/2-1} 2/(n+k) \simeq 4 \cdot \int_0^{n/2} 1/(n+k)\mathrm{d}k = 4 \cdot \ln(3/2) < 1.63$. □

4.2 Non-Simultaneous Scattering

Suppose that the packets that have to be scattered over the rows are not all present by the time the scattering on the packets that have to go farthest has to be performed. Then we have to perform repeated scattering. This costs more time and implies the risk that the quality of the scattering is severely reduced. Sometimes some quality reduction can be buffered, e.g., if spreading is applied, sometimes this is undesirable. We show that the same distribution of the packets over the rows can be achieved as when all packets would be present at the start, if the number of different keys does not exceed s, the size of the submeshes in which the scattering is performed.

Assume that there are exactly s keys $\in [0, s-1]$, and that we want to perform an optimal distribution of the packets with the same keys over the rows. We can make column j responsible for the packets with key j. Suppose that the number x_j of packets with key j that were scattered in this submesh is available in this column. Then, upon arrival of a new charge of packets that have to be scattered, we perform

Algorithm SCATTER

1. Sort the new packets in column-major order.

2. Determine the numbers y_j of packets with key j modulo s.

3. Move the last y_j packets with key j along the rows to column j.

4. Place these last y_j packets within column j in the positions $[(x_j + j) \bmod s, (x_j + j + y_j - 1) \bmod s]$ (cyclically).

5. For all j, $x_j := x_j + y_j$.

The properties of the scattering are resumed in

Lemma 7 *If* SCATTER *is performed t times, there are in total A_j packets with key j, $0 \leq j < s$, and $\sum_j A_j = A$, then at most $\lceil A_j/s \rceil$ packets with key j are placed in any row. The scattering takes $\mathcal{O}(t \cdot s + A/s)$ steps. If in any repetition of* SCATTER *there are in total a packets to scatter then the queue sizes are bounded by* $\lceil a/s^2 \rceil + 1$.

Proof: The time order can be analyzed easily: all steps in SCATTER are simple sort and shift operations. It is essential that the packets are reasonably evenly distributed. The i-th packet with key j is placed in row $(i + j) \bmod s$. This immediately gives the bound on the number of packets with key j that is placed in a row. When a packets are scattered, then they are optimally distributed in Step 1, and then in Step 3 or Step 4 a PU may receive one more packet. □

4.3 Algorithm

With the gathered knowledge about the 'randomization' and the scattering it is now easy to construct a deterministic algorithm with $T = 2 \cdot n + \mathcal{O}(n^{3/4})$ and $Q = 2$.

The algorithm consists alternatingly of local rearrangements and routing phases. It is very similar to BASIC-ROUTE. Most important are the modifications of the pseudo-randomization in Step 2 and the coalescing of the routing phases with the repeated SCATTER in Step 3 and Step 5. The mesh is divided in submeshes of size $s \times s$, and the time in slots of length S. We use $m = n/s$.

Algorithm SMALL-Q-ROUTE

1. In every submesh, sort the packets lexicographically first on the indices of their row-bundles and then on the indices of their column-bundles. For every packet p, standing in row-bundle i, $0 \leq i < m$, with destination in row-bundle k, $0 \leq k < m$, determine the number of row-bundles $a_{i,k}$, that lie within its randomization range. If p has rank r within its submesh, then it gets intermediate destination in row-bundle $r \bmod a_{i,k}$. In every submesh, the packets are sorted in row-major order on the indices of their intermediate destination row-bundles.

2. Route the packets along the columns to the first PUs in their intermediate destination row-bundles holding no other packets of this type.

3. In every $s \times s$ submesh, scatter the packets that just finished Step 2 in column-major order, every S steps.

4. Route the packets along the rows to the first PUs in their destination column-bundles holding less than two packets of this type.

5. In every submesh, scatter the packets that just finished Step 4 in row-major order, every S steps.

6. Route the packets along the columns to the first PUs in their destination submeshes holding no other packets of this type.

7. In every submesh, route the packets to their destinations, every S steps.

4.4 Analysis

We set $s = n^{1-\epsilon}$ and $S = n^{1-\epsilon/2}$, and determine the largest ϵ for which analogues of Lemma 3 hold:

Lemma 8 *If $\epsilon < 1/2$, then Step 2 can be performed as specified.*

Proof: The randomization range of any packet is at least $n/2$. So, the 'probability' that a packet is routed to some submesh $S_{i,j}$, is at most $2 \cdot s/n$. Taking into account the rounding errors, we find that in any submesh there are at most $\lceil (2 \cdot s^2 \cdot s/n + m)/s \rceil$ packets with destination in $S_{i,j}$ in a single column.

Consider now the PUs from which $S_{i,j}$ may be reached by packets finishing Step 2 between routing step t and $t + S$. Because the packets are not delayed during Step 2, these are PUs in at most $2 \cdot \lceil S/s \rceil$ submeshes. Hence, during S steps, $S_{i,j}$ receives at most $d = 2 \cdot \lceil S/s \rceil \cdot \lceil (2 \cdot s^3/n + m)/s \rceil$ packets in any of its columns. Taking $\epsilon < 1/2$ gives $d = \mathcal{O}(n^{1-3/2 \cdot \epsilon}) = o(s)$. □

Lemma 9 *If $\epsilon < 2/5$, then Step 4 can be performed as specified.*

Proof: If the packets would not be delayed during Step 4, we could have proven analogously to the proof of Lemma 8, that during S steps the number of packets received by a submesh is at most $o(s^2)$. However, packets may be delayed, and applying the farthest-first strategy, this may mean that packets which do no have to travel far are 'wiped together' and all arrive at once.

A particularly bad instance arises for the 'horizontal shift over $n/2$: the permutation under which the packet starting in $P_{i,j}$ has destination in $P_{i,(j+n/2) \bmod n}$. Under this permutation, all approximately $2 \cdot \ln 2 \cdot s^2$ packets going to submesh $S_{m/2,m/2}$ arrive at once. Actually this is a worst case: no permutation 'randomizes' more than $2 \cdot \ln 2 \cdot s^2$ packets with destination in a single column-bundle to the same row-bundle (this can be shown analogously to the proof of Lemma 5). Because in Step 2, in every submesh the number of packets with destination in some row-bundle l, which get intermediate destination in a row-bundle may be rounded up m times, the actual number may be larger by m^3. This means that the number d of packets with destination in $C_{m/2}$, in any row of $R_{m/2}$ satisfies $d = 2 \cdot \ln 2 \cdot s + m^3/s + m$. For $\epsilon < 2/5$ this gives $d = 2 \cdot \ln 2 \cdot s + o(s)$. □

Lemma 10 *If $\epsilon < 1/2$, then Step 6 can be performed as specified.*

Proof: Similar to the proof of Lemma 8. The observation we must make is that the packets are hardly delayed: because the packets are more or less randomized within the columns, part of those that have destinations in the highest i rows already stand there after Step 2. This means that the density in the stream through some connection from row $i+1$ to row i never becomes one. Thus, packets can always start Step 6 within s steps, and packets are not wiped together as a result of the farthest-first strategy. Also, because of the 'randomization' the packets within the stream are fairly distributed and it may not happen that the stream mainly consists of packets with destination in a single submesh. Hence, during S steps in any row of a submesh at most $d = \mathcal{O}(s \cdot S/n + m)$ packets are received. Taking $\epsilon < 1/2$ gives $d = \mathcal{O}(n^{1-3/2 \cdot \epsilon}) = o(s)$. □

Summing all the arising queue sizes gives $Q = 1 + 2 + 1 + 1 = 5$. Using Assumption 2 we may forget about the packets that have reached their destinations and get $Q = 4$.

With a simple modification we can obtain $Q = 2$. The PUs are colored white and black as on a chess board: $P_{i,j}$ is colored white if $i+j$ even, otherwise it is colored black. From the analyses of the above lemmas it follows that the number of packets that finish their

Step 2 and Step 4 during S steps in some submesh $S_{i,j}$ is $o(s)$. Only the packets that finish their Step 4 may be numerous, but their number does not exceed $3/2 \cdot s^2$. This makes it possible to correctly modify Step 2, 4 and 6 as follows:

1. Route the packets along the columns to the first *white* PUs in their intermediate destination row-bundles holding no other packets that just finished Step 2 or Step 6.

2. Route the packets along the rows to the first *white* PUs in their destination column-bundles holding no other packets of this type, or to the first *black* PUs holding less than two packets.

3. Route the packets along the columns to the first *white* PUs in their intermediate destination row-bundles holding no other packets that just finished Step 2 or Step 6.

In this way, the packets leave enough room for each other to always find a place.

Theorem 2 *With $\epsilon < 2/5$,* SMALL-Q-ROUTE *performs permutation routing on an $n \times n$ mesh in $2 \cdot n + \mathcal{O}(m + S + s \cdot n/S)$ steps and with maximal queue size two.*

Proof: Step 1 takes $\mathcal{O}(s)$ steps. As a result of Step 3, 5 and 7, the routing is interrupted n/S times for $\mathcal{O}(s)$ steps. A packet may be waiting S steps before it is handled by each of these steps. In Step 2 the packets move without delay as far as they have to go. A packet spends at most n steps in Step 4: consider a packet p moving through some row. By Lemma 5 and Lemma 4, we know that if all packets would start to move at the same time, that then p would reach its destination column within n steps. Actually, some packets already have covered part of their distance by the time p starts Step 4. This certainly does not increase the duration of Step 4 for p beyond n. In Step 6, packets may need s steps to enter the stream, but then move without further delay. □

Actually we do not need $s = n^{1-\epsilon}$ with $\epsilon < 2/5$. It is sufficient to take $s = c \cdot n^{3/5}$ for suitable constant c.

Corollary 1 *With $s = \mathcal{O}(n^{3/5})$ and $S = n^{4/5}$, the modified version of* SMALL-Q-ROUTE *performs permutation routing on an $n \times n$ mesh in $2 \cdot n + \mathcal{O}(n^{4/5})$ steps and with maximal queue size two.*

If in Step 1, the sorting is performed in $n^{3/4} \times n^{3/4}$ submeshes, then s can be taken $\mathcal{O}(n^{1/2})$ and S equal to $n^{3/4}$. The routing time becomes $2 \cdot n + \mathcal{O}(n^{3/4})$.

4.5 Reducing T

In this section we describe how SMALL-Q-ROUTE can be augmented with the idea of critical packets [10]. Critical packets start in one of the $a \times a$ corners and have destination in an oppositely located corner. They do not perform the local scattering routines but are routed without delay towards their destinations, and are routed recursively within designated areas of the mesh. We first describe the routing of the critical packets and discuss later the problems that arise when non-critical packets are routed simultaneously.

For the first T_s steps, during which the non-critical packets are scattered, the critical packets are routed orthogonally to the non-critical packets. After the critical packets moved horizontally for another $a + T_s$ steps, they perform the recursive routing during which they move within the recursive routing region to destinations that correspond to their final destinations. Then they move on horizontally until they hit their destination columns and finally vertically to their destinations. All together this takes $2 \cdot n - 2 \cdot a - 2 + T_{\mathrm{rc}}$, where T_{rc} is the time for the recursive routing. Note that the recursive routing starts at time step $2 \cdot T_s + a$. To guarantee that the regions for the recursive routing are disjoint we must have $2 \cdot T_s + 2 \cdot a \leq n/2$. With $T_s = 5^1\!/\!_2 \cdot n^{4/5}$, this implies that we must choose

$$a \leq n/4 - 5^1\!/\!_2 \cdot n^{4/5}. \qquad (2)$$

The non-critical packets must not disturb the routing of the critical packets. Therefore we modify Step 2 in the basic algorithm, such that unshuffling of the noncritical packets to the lowest and highest $a + T_s$ rows is not allowed. Each packet is 'randomized' as before but now in a range which is smaller by at most $2 \cdot a + 2 \cdot T_s$: the density in the middle rows increases by a factor smaller than $1 + n/(n - 2 \cdot a + 2 \cdot T_s)$. Thus, the density still remains strictly smaller than two, for all sufficiently small a, in particular for $a = o(n)$. The additional delay during Step 4 and Step 6 is already comprised in our estimates. Performing basically the same analysis as before,

Lemma 11 *Let d_1, d_2, d_3 be the distances a non-critical packet p has to travel in Phase 1, Phase 2 and Phase 3, respectively. The phases are finished after $d_1 + \mathcal{O}(s)$, $d_1 + d_2 + \mathcal{O}(S + s \cdot n/S)$, and $d_1 + d_2 + d_3 + \mathcal{O}(S + s \cdot n/S)$ steps, respectively.*

Let c be the constant hidden in the term $\mathcal{O}(S + s \cdot n/S)$. Since for the $d_1 + d_2 + d_3 \leq 2 \cdot n - a$, we must choose $a \geq c \cdot (S + s \cdot n/S)$, in order to get a routing time of less than $2 \cdot n$ steps for the non-critical packets. When $s = 2 \cdot n^{3/5}$ and $S = n^{4/5}$, we can take $a = 3 \cdot c \cdot n^{4/5}$. Comparing with (2), we find that the recursion must stop when

$$n < (12 \cdot c + 22)^5.$$

This number is huge but constant. If eventually n becomes too small, we apply some routing algorithm that assures $Q = 2$. A good choice is the algorithm from [11], which runs in $5^1\!/\!_2 \cdot n$.

During the last local sorting phase of the non-critical packets, the critical packets run orthogonally to the sorting direction, and always move towards the corner. To make this possible the critical packets end the horizontal move towards their destination columns slightly before they have reached them. By this final

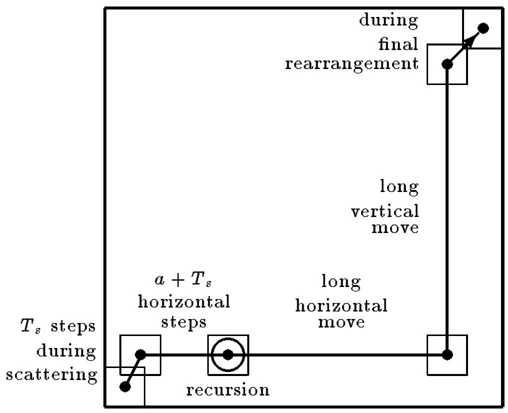

Figure 4: Track of the critical packets starting in the lower-left corner.

provision critical and non-critical packets do never interact and so, the stated running time holds. The complete track of some critical packets is depicted in Figure 4.

Surprisingly, the queue size is 2 as before since the density for the noncritical packets increases only very little and critical and non-critical packets do not interact.

Theorem 3 *Distinguishing critical and non-critical packets,* SMALL-Q-ROUTE *can be modified to perform permutation routing on an $n \times n$ mesh in $2 \cdot n + \mathcal{O}(1)$ steps with maximal queue size 2.*

5 Practical Algorithms

In this section we do not strive for an algorithm with minimal asymptotical running time, but for algorithms which have acceptable run time for *all* n, while bounding Q to a minimum. In this respect BASIC-ROUTE from Section 3 is not too bad, but we want more: we would like to have something like $T \leq 3 \cdot n$, for all n, and $Q < 10$.

5.1 Kunde's Algorithm

The easiest of all algorithms is Kunde's algorithm [5]. The mesh is divided in submeshes of size $s \times s$, and then the following steps are performed:

Algorithm KUNDE-ROUTE

1. Sort the packets in the $s \times s$ submeshes in column-major order.

2. Route the packets along the rows to their destination columns.

3. Route the packets along the columns to their destinations.

Let $T_{\text{sort}}(s)$ be the time for column-major sorting.

Lemma 12 *For* KUNDE-ROUTE $T(n, s) = T_{\text{sort}}(s) + 2 \cdot n - s^2/n$, $Q = 2 \cdot n/s - 1$

Proof: Step 1 takes $T_{\text{sort}}(s)$ steps and Step 3 takes $n - 1$ steps. For Step 2 we notice that after Step 1 a packet that stands in the leftmost column of its submesh apparently has destination in a column with the smallest index of all packets in this submesh. Hence, this packets cannot have destination in any of the rightmost s^2/n columns. Thus, the maximal distance it has to travel rightwards is bounded by $n - s^2/n$. The queue size has been analyzed in [5]. □

Lemma 13 [11] *An $s \times s$ mesh can be sorted in row- or column-major order by a uni-axial algorithm with the routing times T and maximal queue sizes Q given in Table 1.*

s	T	Q
2^l	$5^1\!/\!_2 \cdot s$	2
2^l	$5 \cdot s$	3
2^l	$4^1\!/\!_2 \cdot s$	4
3^l	$4^1\!/\!_3 \cdot s$	5

Table 1: Values of T and Q for uni-axial row-major sorting.

Important is that these results hold for *all s*.

Combining Lemma 12 and Lemma 13 gives

Theorem 4 *Permutation routing on an $n \times n$ mesh can be performed by a uni-axial algorithm with the routing times T and maximal queue sizes Q given in Table 2.*

n	s	T	Q
2^l	$n/2$	$4^1\!/\!_4 \cdot n$	3
$4 \cdot 3^l$	$n/4$	$3^1\!/\!_{48} \cdot n$	7
$8 \cdot 3^l$	$n/8$	$2.53 \cdot n$	15

Table 2: Values of T and Q for uni-axial routing.

Proof: The results are obtained by selecting the appropriate sorting algorithm, and then substituting in the expression of Lemma 12. □

5.2 Reducing Q

In [12] we developed several spreading techniques, to reduce the queue size from $2 \cdot n/s - 1$ down to possibly n/s. A variant of SHORTSPREAD is particularly suited for improving the queue sizes of KUNDE-ROUTE without increasing the routing time.

Consider a packet p walking rightwards to its destination in column j. Let Q be the maximum queue size that we want to allow. Then we apply the following spreading strategy:

Algorithm SHORTSPREAD

Route p to P_j. If P_j holds less than Q packets, then store p in P_j. Else, if P_j holds a packet p' with destination in P_{j+1}, then route p' to P_{j+1} and store p in P_j. Else, route the packet p' that has to leave latest from P_j to P_{j+1}, and store it there.

Notice that SHORTSPREAD may induce a chain reaction. Its most important property is that each packet is stored at most one column away from its destination column. In [12] we prove the correctness of this property for $Q \geq n/s$ (in our case).

We have to make sure that the vertical routing phase is conflict free, namely that each packet p resides in the processor P in its destination column at the moment that it has to start its vertical move. This moment is $T - d_p$ where d_p is the distance packet p has to travel along the column. A problem might arise if two packets were sent simultaneously from processor P, one upwards and one downwards, and in the subsequent steps again. If too many of these packets were stored in one neighbor of P, then they could not move fast enough towards P. Fortunately such a situation cannot occur. Essentially this amounts to the fact that, either at least s steps pass between two successive packets leaving upwards, or sufficiently many packets already stand in P itself. In order to keep the packets that are due to leave ahead, the packets that have to leave first can continuously move back and forth between P, from where they should move vertically, and between its neighbors, until they find a free location in P.

In this way we obtain a queue size n/s, which means a fairly good trade-off between routing times and queue sizes:

Theorem 5 *Permutation routing on an $n \times n$ mesh can be performed by a uni-axial algorithm with the routing times T and maximal queue sizes Q given in Table 3.*

Proof: The spreading has no negative influence on the routing times, so we can still use the expression of Lemma 12. □

n	s	T	Q
2^l	$n/2$	$4^1/_2 \cdot n$	2
$3 \cdot 2^l$	$n/3$	$3^5/_9 \cdot n$	3
2^l	$n/4$	$3^1/_{16} \cdot n$	4
$5 \cdot 3^l$	$n/5$	$2^{62}/_{75} \cdot n$	5
$2 \cdot 3^l$	$n/6$	$2^{25}/_{36} \cdot n$	6
$8 \cdot 3^l$	$n/8$	$2.53 \cdot n$	8

Table 3: Values of T and Q for uni-axial routing with spreading.

5.3 Simulations

For testing the feasibility, correctness and running time in practice, we developed a mesh-simulator system. As software-tool we used the LEDA-system [7, 8]. A special input generator enables the construction of (bad) inputs supported by simple actions with the mouse. The actual simulator supports a rather high-level description of the algorithms, facilitating the step from algorithms on paper to a running simulation. It allows for parallel recursion. A visualisator program shows in a window on the left the current distances the packets still have to go, and in a window on the right the actual queue sizes. In the middle, the maximum distance over all packets is displayed.

We have implemented the enriched version of KUNDE-ROUTE on this simulator. For the sorting we took the uni-axial sorting algorithm SORT-ALL from [11]. This algorithm has a sufficiently simple structure (essentially it is a merge sort), and yet it combines flexibility concerning the side lengths with maximum speed and small queues.

We demonstrate this simulation with an example. For $n = 128$ and $s = n/4$, we consider a permutation under which the packets in the upper 32 rows are rotated symmetrically into the rightmost 32 columns. The packets of the lowest 32 rows are randomly permuted into the leftmost 32 columns. The packets in the central 64×64 square do not have to move, whereas all remaining packets are randomly permuted over the free PUs. Figure 5 (on Page 11) shows the configuration after the local sorting: on the left the distances are shown and on the right it is indicated that the queue size is 1 again. Figure 6 shows the situation at the end of the horizontal move. The packets that started in the upper and lower row-bundles are now concentrated in the corners. The differences between areas where the packets are permuted randomly and those where they are permuted deterministically can be distinguished clearly. Finally, Figure 7 shows a stage during the vertical move of Step 3. In the central square all packets have already reached their final destinations.

n	s	T_{theory}	T_{real}	T_{real}/n	Q
128	$n/4$	392	395	3.09	4
256	$n/4$	784	788	3.08	4
135	$n/5$	382	385	2.85	5
270	$n/5$	764	768	2.84	5
162	$n/6$	437	440	2.72	6
144	$n/8$	364	366	2.54	8

Table 4: Comparison between theory and practice for uni-axial routing with spreading.

For various mesh sizes and choices of s we have obtained the values of T and Q given in Table 4. The slight differences between theory an practice mainly goes back on suboptimal implementations of the sorting algorithms in order to keep them practical. In part the chosen values for n and s do not allow the full application of the fastest version of SORT-ALL. Nevertheless our simulation shows that the claimed bounds are not only correct, but are really attainable by an algorithm of moderate complexity.

References

[1] Chlebus, B.S., M. Kaufmann, J.F. Sibeyn, 'Deterministic Permutation Routing on Meshes,' *Proc. 5th Symp. on Parallel and Distributed Processing*, pp. 814–821, IEEE, 1993.

[2] Kaklamanis, C., D. Krizanc, S. Rao, 'Simple Path Selection for Optimal Routing on Processor Arrays,' *Proc. 4th Symp. Parallel Algorithms and Architectures*, pp. 23–30, ACM, 1992.

[3] Kaufmann, M., J.F. Sibeyn, 'Optimal Multi-Packet Routing on the Torus,' *Proc. 3rd Scandinavian Workshop on Algorithm Theory*, LNCS 621, pp. 118–129, Springer-Verlag, 1992.

[4] Kaufmann, M., J.F. Sibeyn, T. Suel, 'Derandomizing Algorithms for Routing and Sorting on Meshes,' *Proc. 5th Symposium on Discrete Algorithms*, pp. 669–679, ACM-SIAM, 1994.

[5] Kunde, M., 'Routing and Sorting on Mesh Connected Processor Arrays,' *Proc. VLSI Algorithms and Architectures*, LNCS 319, pp. 423–433, 1988.

[6] Leighton, T., F. Makedon, I.G. Tollis, 'A $2n-2$ Step Algorithm for Routing in an $n \times n$ Array with Constant Size Queues,' *Proc. Symp. Parallel Algorithms and Architectures*, pp. 328–335, ACM, 1989.

[7] Mehlhorn, K., Näher St. 'LEDA: A library of efficient data types and algorithms,' *Proc. 14th Symp. on the Mathematical Foundations of Computer Science*, LNCS 379, pp. 88–106, 1989.

[8] Näher St. 'LEDA – Manual,' *Techn. Rep. MPI-I-93-109*, Max-Planck Institut für Informatik, Saarbrücken, Germany, 1993.

[9] Rajasekaran, S., R. Overholt, 'Constant Queue Routing on a Mesh,' *Journal of Parallel and Distributed Computing*, pp. 160–166, June 1992.

[10] Rajasekaran, S., Th. Tsantilas, 'Optimal Routing Algorithms for Mesh-Connected Processor Arrays', *Algorithmica*, 8, pp. 21–38, 1992.

[11] Sibeyn, J.F., 'Desnakification of Mesh Sorting Algorithms,' *Techn. Rep. MPI-I-94-102*, Max-Planck Institut für Informatik, Saarbrücken, Germany, 1994. *Proc. 2nd European Symposium on Algorithms*, LNCS 855, pp 377–390, Springer-Verlag, 1994.

[12] Sibeyn, J.F., B.S. Chlebus, M. Kaufmann, J.F. Sibeyn, 'Shorter Queues for Permutation Routing on Meshes,' *Proc. 19th Symposium on the Mathematical Foundations of Computer Science*, LNCS 841, pp. 597–607, Springer-Verlag, 1994.

Figure 5: Example for $n = 128$. After sorting locally.

Figure 6: After the horizontal routing phase.

Figure 7: After the vertical routing phase.