VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wananga o te Upoko o te Ika a Maui

Department of Computer Science

PO Box 600
Wellington
New Zealand

Tel: +64 4 471 5328
Fax: +64 4 495 5232
Internet: Tech.Reports@comp.vuw.ac.nz

Efficient Token Distribution and Load Balancing on Reconfigurable Meshes with Restricted Bus Length

Heiko Schröder and **Gavin Turner**

Technical Report CS-TR-96/12
July 1996

Abstract

A solution to the token distribution problem is presented for the 2-dimensional reconfigurable mesh with restricted bus length. The algorithm is shown to be asymptotically worst-case optimal in reducing the discrepancy Δ between maximum and minimum processor loads to δ in optimal $\Theta((\Delta - \delta) \cdot n)$ time steps. The algorithm meets the time complexity of current state-of-the-art algorithms for sorting and permutation routing, but remains a factor of 2 from the bisection bound.

Publishing Information

Submitted: Australian Computer Science Conference, ACSC'97, Sydney Australia

Author Information

Heiko Schröder studied mathematics, physics, and computer science at CAU in Kiel (Germany) where he received his Ph.D. in computer science in 1977. Until 1989 he was Assistant Professor of Computer Science at CAU and Kansas University (U.S.A.) and Senior Research Fellow at ANU in Canberra (Australia). Until 1994 he was Professor of Microelectronics in Newcastle (Australia) and is currently Professor of CS in Loughborough (U.K.). His main research area is massively parallel computing including algorithms and architectures for problems related to sorting, image processing and fault tolerance.

Gavin Turner received the Bachelor of Computer Engineering (Hon) and the Bachelor of Computer Science (Hon) degrees at the University of Newcastle, Australia in 1991 and 1992 respectively. From 1993 to 1995 he undertook a Ph.D. program at the same institution, studying the VLSI design and analysis of load balancing systems. His Phd dissertation is currently being accessed. In 1995 he was employed as an associate lecturer in the Department of Computer Science at the University of Newcastle, Australia. He is currently a lecturer in Computer Science at Victoria University of Wellington, Wellington New Zealand. His main research interests lie in the area of design and analysis of algorithms, in particular algorithms and architectures for special-purpose parallel computing. In addition, he is interested in a wide variety of problems from image processing, load balancing and combinatorics.

Efficient Token Distribution and Load Balancing on Reconfigurable Meshes with Restricted Bus Length

Heiko Schröder
H.Schroder@lut.ac.uk
Department of Computer Studies,
University of Loughborough, LE11 3TU, UK

Gavin Turner
gavin@comp.vuw.ac.nz
Department of Computer Science,
Victoria University of Wellington, PO Box 600, Wellington, New Zealand
(Direct all Correspondence to this Author)

Abstract

A solution to the token distribution problem is presented for the 2-dimensional reconfigurable mesh with restricted bus length. The algorithm is shown to be asymptotically worst-case optimal in reducing the discrepancy Δ between maximum and minimum processor loads to δ in optimal $\Theta((\Delta-\delta)\cdot n)$ time steps. The algorithm meets the time complexity of current state-of-the-art algorithms for sorting and permutation routing, but remains a factor of 2 from the bisection bound.

Keywords : Load Balancing, Token Distribution, Reconfigurable Meshes

1 Introduction

One of the fundamental data distribution problems on parallel architectures is that of *token distribution*, a static variant of the well-studied load balancing problem, and akin to the problems of sorting and permutation routing. Each processing element (PE) of the parallel architecture possesses an initial set of *tokens*, each of which represents a task to be performed; the number of tokens stored at a particular PE is called the *load* of that PE. Ideally, one would prefer that the distribution of the tokens over the set of PEs be as even as possible, as imbalances would result in a delay in the time needed to perform all the tasks. The goal of a token distribution algorithm is to redistribute the tokens in such a way that the final loads of the PEs differ by as little as possible. Here it is assumed that each token requires only a constant amount of time to send from one PE to an adjacent PE, and that no tokens are created or destroyed before the redistribution is complete.

One of the simplest and therefore one of the most practical fixed connection networks is the mesh. In this model, the processing elements of the parallel machine are arranged in a square grid, and are connected to their neighbours by short, fixed, uni-directional links. A dynamic variant of the fixed connection mesh, the so-called *reconfigurable mesh*, has become popular with researchers over the last few years [7]. Under this model, which has been shown to have hardware cost comparable with that of the fixed connection mesh [8], the PEs are augmented with a small switching network with which they may locally short connections to form busses. In each instruction step, a message can be sent along a bus. In extreme cases, a row or column consists of only one bus of length in $\mathcal{O}(n)$ (where n is the sidelength of the mesh), or (yet more extreme) row and column busses may be joined to form a single bus of length in $\mathcal{O}(n^2)$.

The popularity of the reconfigurable mesh model is due to the fact that, while it shares many of the advantages of the fixed-connection mesh – it is extremely suitable for VLSI fabrication and therefore massively parallel computing – it removes one of the meshes greatest drawbacks – its large diameter. The reconfigurable mesh has an effective diameter in $\mathcal{O}(1)$. This is gained, of course, at the expense of the meshes *scalability*. Since electrical signals may only travel a certain distance in a certain time (dependent on the technology used), the sidelength of the reconfigurable mesh places a lower bound on the length of an instruction cycle.

In this paper, we consider an SIMD mesh in which the PEs are connected via a reconfigurable mesh topology, but on which we impose some practical assumptions. In the *restricted reconfigurable model*, it is assumed that signals may only travel a certain distance in one instruction cycle – constraining the length of any busses which may be formed by the algorithm. We measure this length l as an arbitrary, integer number of fixed connections. The result of imposing such an assumption is that the scalability of the architecture is restored - the length of an instruction cycle is bounded by the constant l and not n. The cost, however, is that the diameter of the architecture is increased from $\mathcal{O}(1)$ to $\mathcal{O}(n/l)$.

Many solutions to the token distribution problem have been proposed for a variety of interconnection networks [10, 9, 11, 3]. In 1993 Meyer auf der Heide, Oesterdiekhoff and Wanka [10] introduced an asymptotically optimal algorithm for the token distribution problem on fixed-connection, d-dimensional meshes. This work was followed in 1995 with an algorithm [3] for reconfigurable d-dimensional meshes which outperformed the previous algorithm for token size $> 4\times$ bus bandwidth. There were two major drawbacks with this algorithm. Firstly, as just noted, outperforming the fixed connection algorithm of Meyer auf der Heide, Oesterdiekhoff and Wanka was dependent on the ratio of token size to bus bandwidth. This was satisfactory for load-balancing applications where the token can be assumed to be of considerable size, or over networks in which the communication is bit serial, but neither of these conditions is true in the general case.

The second problem was that there existed a significant difference between the state-of-the-art with respect to solutions for the problems of sorting and permutation routing on reconfigurable arrays, and that of solutions to the token-distribution problem. The problem of reducing the discrepancy Δ (the difference between the maximum number of tokens in any PE and the minimum number of tokens in any PE) of an arbitrary token distribution by 1 is in some sense a *simpler* problem than the problems of sorting and permutation routing since token distribution implies a smaller bisection bound; to decrease Δ by 1, we must simply remove a token from all PEs with the maximum number of tokens OR send a token to all PEs with the minimum number. In terms of bisection bounds, a bound of n exists for the problems of permutation routing and sorting, whilst a bisection bound of $\frac{n}{2}$ exists for decreasing the discrepancy of a token distribution by 1.

Turner *et. al.* [4] attempted to address these problems, proposing an alternate algorithm for the 2-dimensional reconfigurable mesh. This algorithm was shown to decrease the global discrepancy Δ between PE loads by 1 in time $n/2 + o(n)$, therefore meeting the bisection bound. This result, however, was gained under the assumption that the minimum number of tokens in any PE was greater than 0. This is not true, of course, in the general case.

In 1996, Houle *et. al.* [2] proposed an asymptotically optimal algorithm for the fixed-connection mesh. This algorithm improved on the fixed-connection algorithm of Meyer auf der Heide, Oesterdiekhoff and Wanka [10] in that it was capable of reducing Δ by arbitrary amount c, $1 \leq c \leq \Delta - 1$, (the former being capable of reducing Δ to 1 *only*),and was extremely simple to implement. The ability to reduce Δ by 1 on an $n \times n$ fixed connection mesh is a necessary subroutine for the approach introduced in this paper.

To date, no algorithms have been proposed for token distribution under the restricted reconfigurable model. At the time of preparing this manuscript, algorithms exist [5] which solve the problems of permutation routing and sorting on $n \times n$ reconfigurable meshes in time $n + o(n)$ where the lower order terms are of significant order (for example $\mathcal{O}(n^{2/3})$ in the case of [5]). Algorithms also exist [1] which solve the problems of permutation routing and sorting in time $n + o(n) + \mathcal{O}(n/l)$ on a reconfigurable mesh with restricted bus length. In this paper we propose an algorithm which solves the token distribution problem on the reconfigurable mesh with restricted bus length for arbitrary Δ and minimum load $\mu \geq 0$.

The algorithm matches the current state-of-the-art for permutation routing and sorting when reducing Δ by 1, requiring time $n = o(n)$. This is a factor of 2 below the previous reconfigurable algorithm of Turner *et. al.* [4]. However, the algorithm proposed in this paper is applicable to arbitrary token distributions and therefore represents the current state-of-the-art for reconfigurable architectures.

The organization of the paper is as follows: in the next section, we describe the model of computation. In Section 3, we formalise the token-distribution problem and prove a lower bound on its complexity for 2-dimensional mesh-connected architectures. The notation and preliminary concepts that we use in the statement and analysis of the algorithm is introduced in Section 4. The statement and analysis of the algorithm appears in Section 5. Concluding remarks are made in Section 6.

2 Model of Computation

We denote by $M_{n,l}$ the $n \times n$ reconfigurable, 2-dimensional mesh with side length n and bounded bus length l (l-restricted rmesh). We assume that each dimension (side) of the mesh is indexed $0, \cdots, n-1$. A processor is identified as $P_{i,j}$ with i being the row index, and j the column index. Each processor (except for those at the boundary) has 4 neighbouring processors, and the processors are connected via a reconfigurable bus system consisting of a fixed connection network (FNET) and n^2 identical switching networks (RNETs) (one per processor).

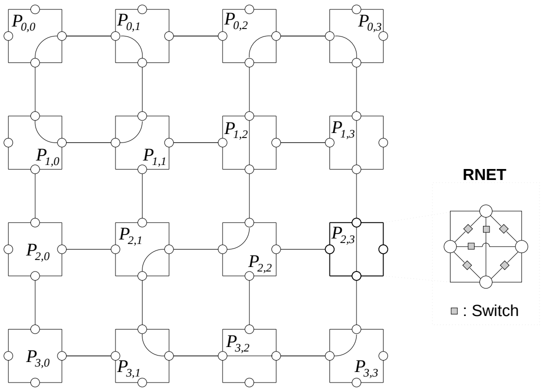

Figure 1: Pictorial Representation of the 2-dimensional Reconfigurable Mesh

Each processor has 4 *ports*. The FNET connects each processor to its 4 neighbouring processors by connecting pairs of ports. The standard 4 neighbourhood notation N,S,E,W is used to denote individual ports. The array of processors together with the FNET is identical to a SIMD fixed connection mesh. An RNET is a switching network present inside each processor which is connected to the 4 FNET ports of the processor. The processor can dynamically configure this switching network so as to electrically short the incoming ports, meaning that any incoming signal on one port will be transmitted to the ports with which it is shorted. Each processor can locally control the bus system using these switches and can write onto or read from any one of the ports.

The configuration of the RNET within each processor will be specified by grouping together ports which have been shorted. Thus if in a processor all ports are electrically isolated, its RNET configuration is denoted as (N,S,E,W). Similarly if only the north and south ports are shorted, the RNET is said to be configured (NS,E,W). A reconfigurable mesh might support all possible shortings. Some examples of these shortings along with their pictorial representation are shown in Figure 2. Note that under the reconfigurable model, busses can be formed which may not be simple (linear) chains of connections. Busses may be *tree-like* (having branches) or even *mesh-like* in structure. Therefore, the *diameter* of a particular bus denotes the longest, shortest distance between any two processors attached to the bus.

We assume that the propagation time for signals along the bus is restricted by the instruction cycle time. We make no restrictions on the way in which processors may locally configure their switching

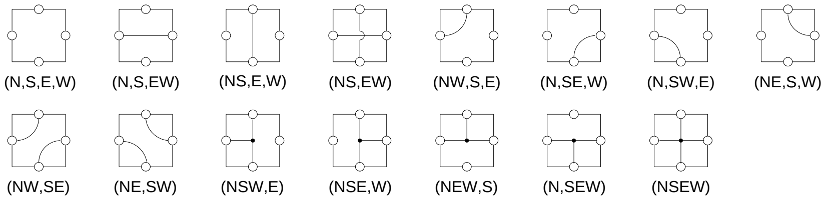

Figure 2: Examples of possible 2-Dimensional switch configurations, and their pictorial representation

networks to form busses, but assume that for busses of diameter greater than l FNET connections, a signal broadcast on that bus cannot be guaranteed to successfully reach any of the PEs to which it is connected (due, for example, to the larger pull-up times required for longer busses). It is the responsibility of any algorithm to maintain that no busses exceed this length, and therefore that no data transfer operation is compromised.

We assume an SIMD architecture where in each atomic instruction a PE:

a) *configures its local RNET,*

b) *writes data to any one port register,*

c) *reads from any one register, including I/O (port) registers,*

d) *performs an elementary calculation,*

e) *writes data to an internal register.*

For each of its 4 ports, a PE contains a dedicated I/O register. It is assumed that the result of the write phase is that the contents of the respective I/O register of the writing PE is replicated in all I/O registers connected to the bus. We assume CREW buses which means that if at any time 2 PEs attempt to write to the same length of bus, the contents of all I/O registers connected to that bus will be corrupted and an error flag will signal. Again it is the responsibility of any algorithm to ensure that CREW bus access is observed.

3 The Token Distribution Problem

The token distribution problem $TD(A;\Delta,M,\delta)$ was first posed by Peleg and Upfal in [9] and can be stated as follows: Given parallel architecture A containing n^2 processors $P_1,...,P_{n^2}$, with each processor P_i containing a stack of $\mu \leq l(P_i) \leq M$ tokens (for all $1 \leq i \leq n^2$) and for a global discrepancy between loads equal to $\Delta = M-\mu$, distribute the tokens such that at the end the global discrepancy has been reduced to at most δ.

For a any 2-dimensional mesh connected architecture, a lower bound on the time required to solve token distribution problem $TD(A;\Delta,M,\delta)$ is easily derived. The result is stated in Lemma 1.

Lemma 1 *Let A be an n-by-n mesh-connected architecture with or without toroidal connections. Any solution to problem* $\mathrm{TD}(A;\Delta,M,\delta)$ *requires time in* $\Omega((\Delta-\delta)\cdot n)$.

Proof The proof is a simple bisection-width argument. Assume a worst-case load in which the $n^2/2$ PEs in one half of A contain M tokens, and the remaining $n^2/2$ PEs contain μ tokens. To reduce Δ to δ, $\frac{\Delta-\delta}{2}\cdot\frac{n^2}{2}$ tokens must be moved from the PEs with M tokens to the PEs with μ tokens. Given that the bisection width of A is $\mathcal{O}(n)$, this requires at least $\Omega((\Delta-\delta)\cdot n)$ steps. □

4 Notation and Preliminaries

The algorithm described in this paper is based on a general method first proposed by Kunde [6] for sorting on fixed-connection meshes. The method makes use of a block-wise decomposition of the $n \times n$ rmesh $M_{n,l}$ into blocks $B_{i,j}$, $0 \leq i, j < n^{1/3}$, each of sidelength $n^{2/3}$ such that block $B_{i,j}$ contains processors at locations $(i \times n^{2/3})$ to $(i \times n^{2/3}) + n^{2/3}$ in rows $(j \times n^{2/3})$ to $(j \times n^{2/3}) + n^{2/3}$. Such a decomposition is shown in Figure 3(a).

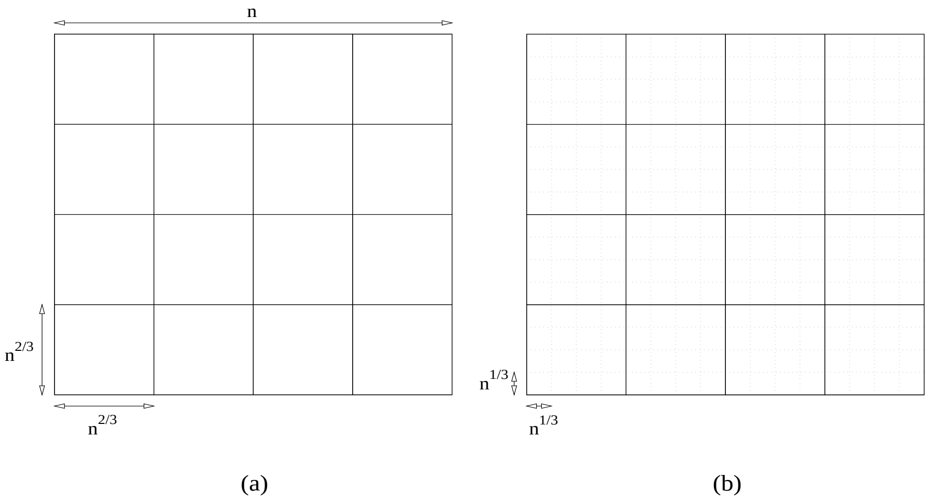

Figure 3: Blockwise Decomposition of the Reconfigurable Mesh

Any block $B_{i,j}$ may be similarly divided into $n^{2/3}$ subblocks $s_{i,j}^{a,b}$, $0 \leq a, b < n^{1/3}$. This further decomposition of the mesh is shown in Figure 3(b). The sorting algorithm of Kunde [6] relies on the implementation of an *All-to-All Mapping* – a process whereby the elements of each block are evenly distributed amongst all other blocks. Specifically, an all-to-all mapping results in subblock $s_{i,j}^{a,b}$ of block $B_{i,j}$ being moved to subblock $s_{a,b}^{i,j}$ of block $B_{a,b}$ for all $0 \leq i, j, a, b < n^{1/3}$. Kunde *et al.* [1] have shown that an all-to-all mapping may be implemented in exactly $n/2$ steps on an $n \times n$ reconfigurable mesh with restricted bus length of $l \geq 8$ units or more. The ability to perform efficient all-to-all mapping (that is in exactly n/2 steps) is an essential element of the algorithm presented here, and for a detailed analysis of this procedure the reader is directed to Kunde *et. al.* [1].

In the course of describing the algorithm, it will be necessary to refer to both reconfigurable meshes with restricted bus lengths, and standard fixed-connection meshes. Unless otherwise specified, the term *l-restricted rmesh* will be used to denote $M_{n,l}$, and *mesh* will be used to denote a fixed connection mesh.

5 Algorithm and Analysis

In this section we present our algorithm, RestrictedBalance which reduces the discrepancy Δ by 1 on the l-restricted rmesh in at most $n + 4n/l + o(n)$ token transfer steps, leading to an overall worst-case time complexity of $\mathcal{O}((\Delta - \delta)(n + 4n/l))$ steps for problem $TD(M_{n,2}; \Delta, M, \delta)$. The algorithm operates in a manner similar to the well known sorting algorithm of Kunde [6], which partitions the mesh into blocks of sidelength $o(n)$ and sorts by performing local sorting steps over these smaller blocks and at particular time instants, evenly distributing the contents of each block over the set of all blocks.

The algorithm which we propose in this paper is formally stated in Figure 4 and the details are discussed in subsequent sections. The algorithm is illustrated in Figure 5 and, informally, operates as follows. As in the algorithm of Kunde [1], the mesh is partitioned into blocks of size $o(n)$ (as shown in Figure 3(a)). Initially, the token distribution algorithm of [2] for fixed-connection meshes is executed in each block to locally remove either all maxima or all minima (Figure 5 pane 1). To remove the remaining maxima and minima a matching process is undertaken whereby tokens from maxima are matched with the addresses of minima.

This matching process is completed as follows. A single token is chosen from each remaining maxima, and for each remaining minima a packet is generated containing the address of the minima. The set of tokens or addresses from each block is evenly distributed amongst all other blocks (Figure 5 pane 2). Within each block, tokens are matched to addresses, and then routed to the corresponding minima (the minima identified in the matched address) (Figure 5 pane 3). In a final clean up step, tokens and addresses which remain are routed to a central block (Figure 5 pane 4) and the matching process is repeated (Figure 5 pane 5). The result is that either all tokens have been matched (all maxima loose a token) or all addresses are matched (all minima receive a token). The algorithm is described in more detail in the following sections.

1 **Initialisation**

- 1.1 Locally balance all blocks.
- 1.2 Minima at mesh location (i, j) generate an address packet containing (i, j) and maxima at location (a, b) select one token to be transmitted. (These are collectively referred to as the *elements*)

2 **Remove Most**

- 2.1 Distribute the elements of each block $B_{i,j}$, $0 \leq i, j < n^{1/3}$, such that the elements in column c, $0 \leq c \leq n^{2/3}$ of block $B_{i,j}$ are moved to block $B_{(c \bmod n^{2/3}),(c/n^{2/3})}$.
- 2.2 In each block $B_{i,j}$, $0 \leq i, j < n^{1/3}$, match all tokens to a unique address such that either all tokens or all addresses are matched (whichever is the lesser).
- 2.3 Route matched tokens back to the PE named in their matched address. For each matched address, a minima will be removed from the distribution.

3 **Clean Up**

- 3.1 Gather the remaining tokens and addresses into central block $B_{(n^{1/3}/2),(n^{1/3}/2)}$.
- 3.2 Match the remaining tokens to an address or addresses to a token, whichever is the lesser.
- 3.3 Route matched tokens back to the PE named in their matched address.

Figure 4: Algorithm RestrictedBalance

5.1 Phase 1: Initialisation

The purpose of the initialisation phase is to firstly remove all maxima and minima within the same block (step 1.1), and then to prepare tokens from maxima, and address packets from minima for their distribution to all other blocks (step 1.2 Figure 4).

For a mesh of sidelength n, algorithms exist [2] which can reduce Δ by 1 in time $\mathcal{O}(n^{2/3})$ on an $n^{2/3} \times n^{2/3}$ mesh. Step 1.1 may be easily implemented then in time $\mathcal{O}(n^{2/3})$ using such an algorithm, the result being that processors in each block have at most $M-1$, or at least $\mu+1$ tokens. In step 1.2, each minima generates a data packet containing its unique mesh address. At the same time, each maxima selects a single token to be sent to a remaining minima. Each address packet indicates a unique (final) destination for a single token.

In total, the number of steps required for Phase 1 of algorithm RestrictedBalance is in $\mathcal{O}(n^{2/3})$.

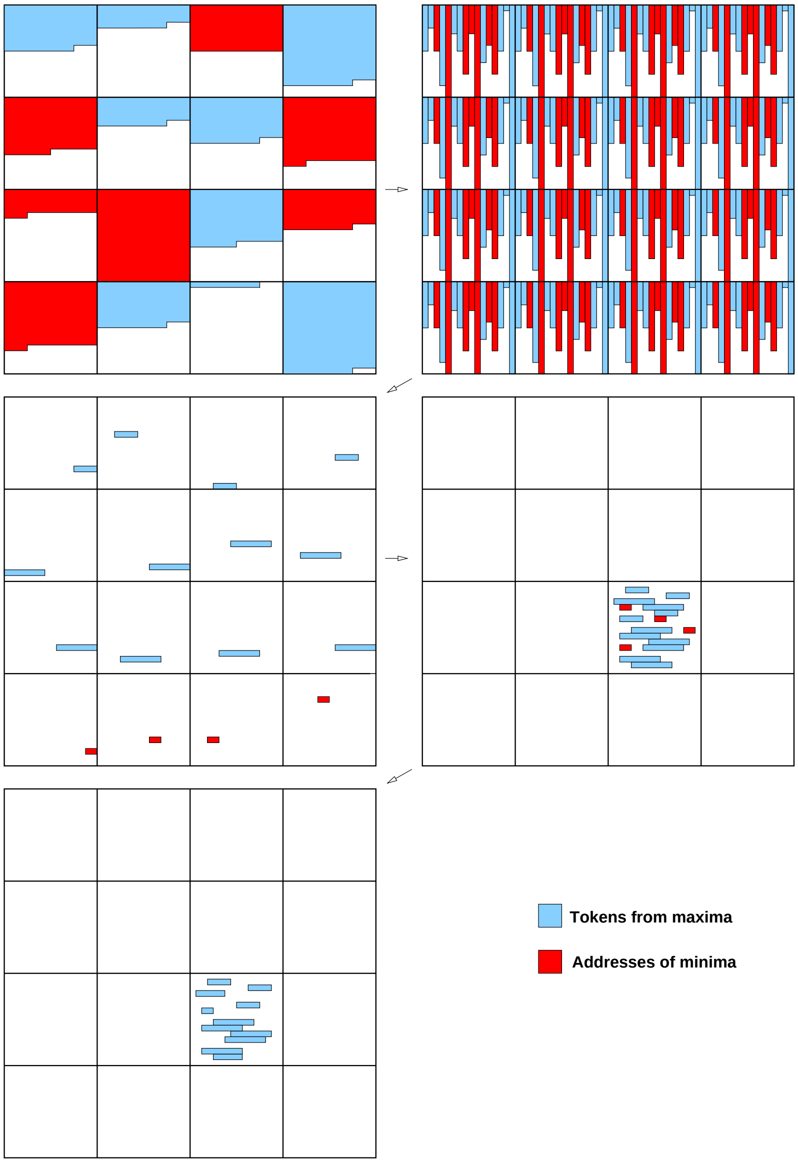

Figure 5: Illustrating the Operation of RestrictedBalance

5.2 Phase 2: Remove Most

The ability to distribute $n^{2/3}$ data elements from each of $n^{2/3}$ blocks to all others has been a necessary component of many of the fastest reconfigurable mesh algorithms [6, 5, 4]. In each of these cases, the algorithm to do so has taken the following general form:

Blockwise_Globally_Distribute

1 In each block $B_{i,j}$ pack the $n^{2/3}$data elements destined for block $B_{a,b}$ into subblock $s_{i,j}^{a,b}$. This requires time $\mathcal{O}(n^{2/3})$, even on an $n^{2/3} \times n^{2/3}$ mesh [12].

2 Produce an *All-to-all Mapping* of subblocks such that the contents of subblock $s_{i,j}^{a,b}$ is routed to subblock $s_{a,b}^{i,j}$.

Kauffman *et. al.* [5] showed that step 2 of the Blockwise-Distribution process, the so called *All-to-all Mapping* could be implemented in exactly $n/2$ steps on an $n \times n$ reconfigurable mesh with no bus restrictions. In addition, Kunde *et. al.* [1] made the spectacular discovery that for l-restricted rmeshes where $l \geq 8$, the all-to-all mapping could be implemented in exactly the same number of steps! This fact was proven and subsequently used in their study of sorting algorithms for l-restricted rmeshes. The algorithm presented in this paper also requires the use of an all-to-all mapping facility. In this paper, we adopt the method proposed by Kunde, and the reader is referred to [1] for specific details of the algorithm.

Algorithm RestrictedBalance requires that the tokens and addresses of each block be evenly distributed amongst all blocks. With this in mind, step 2.1 of RestrictedBalance may be implemented in $n/2+\mathcal{O}(n^{2/3})$ steps on $M_{n,l}$ using the following subroutine:

2.1 Evenly_Distribute

1 In each block $B_{i,j}$ pack the tokens or addresses into the upper left positions of the block (row major ordering). This can be done using a packing algorithm [12] in time $\mathcal{O}(n^{2/3})$ on an $n^{2/3} \times n^{2/3}$ mesh.

2 In each block $B_{i,j}$ route the at most $n^{2/3}$ tokens/addresses in column c into subblock $s_{i,j}^{c \bmod n^{1/3}, c/n^{1/3}}$. This again requires time $\mathcal{O}(n^{2/3})$, even on an $n^{2/3} \times n^{2/3}$ mesh using a suitable routing algorithm [12].

3 Produce an *All-to-all Mapping* of subblocks such that the contents of subblock $s_{i,j}^{a,b}$ are routed to subblock $s_{a,b}^{i,j}$.

Given that all unmatched tokens and addresses can be evenly distributed amongst blocks, it remains for us to describe firstly how tokens and addresses are matched within the blocks (step 2.2), and secondly how the tokens are routed back to their assigned minima (step 2.3 Figure 4).

To implement step 2.3, that is the routing of each matched token back to its addressed minima, we (again) use the all-to-all mapping of Kunde *et al* to move each token to the correct block, and then use a local routing algorithm within each block to finish the job. For this scheme to work properly, all tokens from block $B_{i,j}$ whose final destination address is in block $B_{a,b}$ must reside in subblock $s_{i,j}^{a,b}$ prior to commencing the second all-to-all mapping. With this in mind, we propose the following mechanism to match tokens to addresses and move them to the correct subblock ready for routing to their assigned destination.

2.2 Blockwise_Match

1 Each processor containing an address packet, appends it's unique address to the address packet (denoted the *temporary address*).

2 In each block $B_{i,j}$ pack all tokens into the upper left positions of the block (row major ordering). This requires time $\mathcal{O}(n^{2/3})$ as above [12].

3 Repeat step 2 for addresses.

4 Each processor containing both a token and an address attaches the token to the address packet (the token has been matched)

5 In each block, all matched tokens are routed back to the temporary address contained in their address packet. This requires time $\mathcal{O}(n^{2/3})$ as above [12].

With all matched tokens destined for block $B_{a,b}$ moved to subblocks $s_{i,j}^{a,b}$ for all $0 \leq i,j < n^{1/3}$, step 2.3 can be easily implemented as outlined above, using an all-to-all mapping and then locally routing all matched tokens to their final destination in each block. The resulting subroutine is given below.

2.3 Route_Matched_Tokens

1 Produce an *All-to-All Mapping* of subblocks such that the contents of subblock $s^{a,b}_{i,j}$ is routed to subblock $s^{i,j}_{a,b}$. This requires exactly $n/2$ communication steps [1].

2 In each block $B_{i,j}$ route all matched tokens to their final destination as stored in their address packet. This can be accomplished in time $\mathcal{O}(n^{2/3})$, even on an $n^{2/3} \times n^{2/3}$ mesh using a suitable routing algorithm [12].

In total, the number of steps required for Phase 2 of algorithm RestrictedBalance is $n + \mathcal{O}(n^{2/3})$.

5.3 Phase 3: Clean Up

Phase 3 of algorithm RestrictedBalance removes those tokens or addresses (whichever is the lesser) which remain as a result of phase 2. Phase 3 is necessary since the uniform distribution of tokens and addresses amongst blocks performed in phase 2 (step 2.1 of Figure 4) incurs an inherent error of up to $n^{2/3}$ extra tokens and $n^{2/3}$ extra addresses between blocks (each block contributes an error of at most 1 token/address). Lemma 1 formally characterises the distribution of unmatched tokens and addresses which exist at the end of phase 2.

Lemma 2 *If after Phase 2 of algorithm* RestrictedBalance *there exist both unmatched tokens and unmatched addresses, then the number of unmatched elements per block is at most $n^{2/3}$.*

Proof We know that in phase 2, each block receives a number of tokens equal to the average number across the entire mesh, plus at most $n^{2/3}$extra. Similarly each block receives a number of addresses equal to the average number across the entire mesh, plus at most $n^{2/3}$extra. It follows that if the average number of tokens is at least $n^{2/3}$ more than the average number of addresses, then no addresses will remain *in any block* after phase 2. Similarly, if the average number of addresses is at least $n^{2/3}$ more than the average number of tokens, then no tokens will remain in any block after phase 2. This means that if the number of unmatched tokens remaining in any block after phase 2 is greater than $n^{2/3}$ then there cannot exist unmatched addresses in any other block. Therefore, if there exist both unmatched tokens and unmatched addresses after phase 2, the number in each block must be at most $n^{2/3}$. □

Lemma 1 gives us that after phase 2, the number of unmatched tokens and addresses is at most $(n^{2/3})^2$, being the area of a single block. Therefore, phase 3 performs the final matching by firstly moving all unmatched elements to a single block, then repeating the matching process of phase 2, and then finally routing the matched tokens to their destination. Moving the unmatched tokens to this single central block (step 3.1 of Figure 4) can be accomplished in at most $\mathcal{O}(n^{2/3}) + 2n/l$ on a l-restricted rmesh using the method given below.

3.1 Gather_UnMatched

1 Identify the first $n^{2/3}$ remaining tokens or address in each block. This can be accomplished using a simple enumeration in time $\mathcal{O}(n^{2/3})$ on an $n^{2/3} \times n^{2/3}$ rmesh [3].

2 In each block $B_{i,j}$ route these at most $n^{2/3}$ tokens or identifiers to subblock $s^{i,j}_{i,j}$. This can be accomplished in time $\mathcal{O}(n^{2/3})$ on an $n^{2/3} \times n^{2/3}$ mesh [12].

3 Perform an *in-place move* of all subblocks $s^{i,j}_{i,j}$, $0 \leq i,j < n^{1/3}$, to subblocks $s^{i,j}_{(n^{1/3}/2),(n^{1/3}/2)}$ respectively. This requires exactly $2(n^{2/3} + n/l)$ steps and shall be described below.

The *in-place movement* of subblocks $s^{i,j}_{i,j}$, $0 \leq i,j < n^{1/3}$, to subblocks $s^{i,j}_{(n^{1/3}/2),(n^{1/3}/2)}$ respectively is a non-trivial task which has not been previously investigated in the literature. For this reason, we present the algorithm for performing this move on the l-restricted rmesh, as well as the corresponding analysis, in a separate section - section 5.4 - below.

Once the unmatched tokens and addresses have been gathered in block $B_{(n^{1/3}/2),(n^{1/3}/2)}$, step 3.2 of algorithm RestrictedBalance matches them in exactly the same manner as described in step 2.2. The only difference being that in step 5 of Blockwise_Match, tokens with destination address in block $B_{i,j}$ are routed back to subblock $s^{i,j}_{(n^{1/3}/2),(n^{1/3}/2)}$. This is in preparation for the final route which begins with a second (but opposite) in-place move of the subblocks of block $B_{(n^{1/3}/2),(n^{1/3}/2)}$ back to their original position. The implementation of the final route is formally stated below.

3.3 Final_Route

1 Perform an *in-place move* of subblocks $s^{i,j}_{(n^{1/3}/2),(n^{1/3}/2)}$, $0 \leq i, j < n^{1/3}$, to subblocks $s^{i,j}_{i,j}$ respectively. This requires exactly $2(n^{2/3} + n/l)$ steps and is the exact opposite operation as that performed in Gather_UnMatched above, requiring exactly $2(n^{2/3} + n/l)$ steps.

2 In each block $B_{i,j}$ route all matched tokens to their final destination address. This can be accomplished in time $\mathcal{O}(n^{2/3})$, even on an $n^{2/3} \times n^{2/3}$ mesh using a suitable routing algorithm [12].

Given that the in-place movement of subblocks can be carried out in at most $\mathcal{O}(n^{2/3}) + n/l$ steps (the procedure is described below), phase 3 of algorithm RestrictedBalance can be implemented in $\mathcal{O}(n^{2/3}) + 4n/l$ steps.

5.4 In Place Movement of Subblocks

In this section we present an algorithm for the following subproblem - the movement of data elements from all subblocks $s^{i,j}_{i,j}$, $0 \leq i, j < n^{1/3}$ to subblocks $s^{i,j}_{(n^{1/3}/2),(n^{1/3}/2)}$ respectively on the l-restricted rmesh. The method we propose performs the required move in $\mathcal{O}(n^{2/3}) + 2n/l$ steps. The method first moves the data elements along rows to the correct column (Phase 1) and then along columns to the correct row (Phase 2). The two phases are identical, and it suffices for us to describe the move in one direction only. Algorithm In_Place_Move given below moves the data elements from each row to the correct column in $\mathcal{O}(n^{2/3}) + n/l$ steps using the concept of *stations*.

In_Place_Move

1 In each row all PEs (i, j), where $i \bmod \lfloor n/l \rfloor = 0$ or $i = n - 1$, label themselves as *stations*.

2 All PEs which are the source of a data element or the destination for a data element also label themselves as *stations* (if not already).

3 Repeat until all data elements have reached their destination

3.1 all stations cut the row bus and attempt to read a data element from the West. all stations which contain a data element which needs to travel to the right write that data element to their Eastern ports.

3.2 all stations cut the row bus and attempt to read a data element from the East. all stations which contain a data element which needs to travel to the left write that data element to their Western ports

3.3 if a station receives a data element for which it is the destination, then it stores that element.

The stations identified in the first step of the algorithm are shown in the top illustration of Figure 6. There are at most $n/l+1$ such stations. The stations identified in the second step of the algorithm are shown in the bottom illustration of Figure 6. There are at most $2(n^{1/3}\times n^{1/3})$ such stations since the width of a subblock is $n^{1/3}$ and there are $n^{1/3}$ subblocks which are the source of data elements. Therefore, since there are at most $2(n^{1/3}\times n^{1/3})+n/l+1$ stations along a row, and since no two stations can be more than distance n/l apart, all data elements reach their destination in at most $2(n^{1/3}\times n^{1/3})+n/l$ time steps.

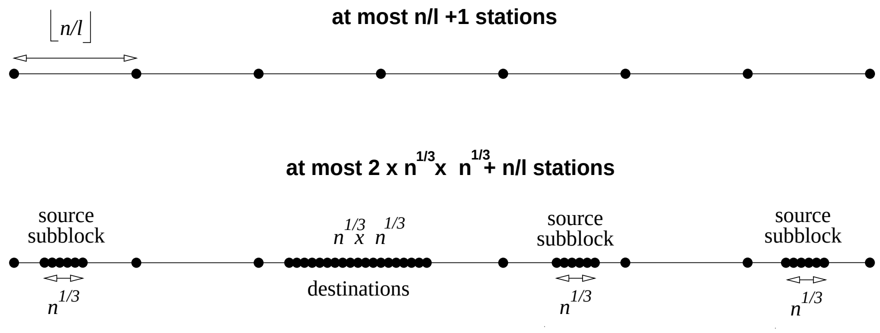

Figure 6: Establishing Row Stations for Moving Data Elements Horizontally

6 Conclusion

In this paper we have introduced a new algorithm for Token Distribution on $n\times n$ meshes. To reduce the global discrepancy of any token distribution problem by 1, the algorithm requires time in $n+o(n)$. The algorithm therefore provides a solution to the general token distribution problem $TD(A;\Delta,M,\delta)$ with time complexity in $\mathcal{O}((\Delta-\delta)\cdot n)$ and is therefore worst-case optimal. For $n\times n$ fixed-connection meshes, the algorithm requires $5n+o(n)$ steps to reduce Δ by 1. Furthermore, on the $n\times n$ reconfigurable mesh with unrestricted bus length, the algorithm requires $n+o(n)$ steps, meeting the time complexity of state-of-the-art algorithms for sorting and permutation routing for the same architecture. Lastly, the algorithm requires $n+4n/l+o(n)$ steps on an l-restricted reconfigurable mesh, simply incurring an additive factor equivalent to the diameter of the network.

A few areas of future work remain. Firstly, the algorithm presented here is *worst-case* optimal. That is, it comes close to the bisection bound of $n/2$ for worst-case distributions which have large numbers of maxima and minima. For more *sparse* distributions however, an algorithm which meets the diameter bound of n/l would be more desirable. The authors are currently investigating methods for sparse distributions.

References

[1] Kunde, M. and Gürtzig, K., 'Efficient Sorting and Routing on Reconfigurable Meshes Using Restricted Bus Length', *In Submission.*

[2] Houle, M. E. and Turner, G., 'Dimension-Exchange Token Distribution on the Mesh and Torus', *Technical Report CS-TR-96/8*, Department of Computer Science, Victoria University of Wellington, New Zealand.

[3] Turner, G. and Schröder, H., 'Token Distribution on Reconfigurable d-Dimensional Meshes', *Parallel Algorithms and Applications*, Vol 8, 1996, pp 251-269.

[4] Turner, G. and and Schröder, H., 'Fast Token Distribution on Reconfigurable Meshes', Proceedings of the 2nd Australasian Conference on Parallel and Real Time Systems, 1995, pp127-133.

[5] Kaufmann, M., Schröder, H., and Sibyen, J., 'Fast Sorting and Routing on Reconfigurable Meshes', *Parallel Processing Letters*, Vol 5, 1995, pp81-95.

[6] Kunde, M., 'Block Gossiping on Grids and Tori: Deterministic Sorting and Routing Match the Bisection Bound', In *Proceedings of the 1st European Symposium on Algorithms*, pp272-283, 1993.

[7] J.W. Jang, H. Park and V.K. Prasana, 'A bit model of the reconfigurable mesh', *In Proceedings of Reconfigurable Architecture Workshop.* IEEE, April 1994.

[8] Lie, K.T., Schröder, H., 'A Reconfigurable Mesh Processor Array Utilising Fault-Tolerant Network', *PRFTS 93*, Melbourne, December 1993.

[9] Peleg, D., Upfal, E., 'The Generalised Packet Routing Problem', *Theoretical Computer Science*, 53, pp218-293, 1987.

[10] Meyer auf der Heide, F., Oesterdiekhoff, B., Wanka,R., 'Strongly Adaptive Token Distribution', *Proceedings of the 20th ICALP*, pp398-409, 1993.

[11] C. G. Plaxton, 'Load balancing, selection and sorting on the hypercube', In *Proceedings of the ACM-SPAA*, pp64-73, 1989.

[12] Leighton, F.T., 'Introduction to Parallel Algorithms and Architectures: Arrays, Trees and Hypercubes', Morgan Kaufmann Publishers, San Mateo California, USA.