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

Optimal Dimension-Exchange Token Distribution on Complete Binary Trees

Michael E. Houle, **Ewan Tempero**
and **Gavin Turner**

Technical Report CS-TR-97/5
May 1997

Abstract

Load balancing on a multi-processor systems involves shifting work around the system so that each processor has about the same amount of processing to perform. The *token-distribution* problem is a static variant of the load balancing problem for the case in which the workloads in the system cannot be divided arbitrarily; that is, where each token represents an atomic element of work. A simple, scalable method for distributing tokens over a distributed-memory parallel architecture is the so-called *dimension-exchange* approach, which is based on the repetitive application of an extremely simple and scalable local exchange protocol. The behaviour of this approach depends on the topology of the interconnection network.

This paper presents the analysis of dimension-exchange algorithms for token distribution on the complete binary tree. We show that for the height H complete binary tree and any initial distribution for which the discrepancy in workloads is greater than H tokens, the dimension-exchange approach leads to the eventual convergence of the distribution such that the discrepancy is at most H. Furthermore, we show that the rate of this convergence is optimal with respect to the bisection width lower bound. These results are the first to establish that dimension-exchange techniques lead to optimal algorithms for finitely-divisible load balancing on a tree-connected network.

Author Information

Michael Houle received the Ph.D. degree in Computer Science from McGill University (Montréal, Québec, Canada) in 1989. After almost three years as a research associate at Kyushu University and the University of Tokyo in Japan, in 1992 he joined the University of Newcastle (NSW, Australia) as a lecturer in the Department of Computer Science and Software Engineering. His main research interests lie in the area of design and analysis of algorithms, in particular computational geometry, combinatorial algorithms, graph drawing, and combinatorial geometry.

Ewan Tempero received a Ph.D. in Computer Science from the University of Washington in 1990, and is currently is a lecturer in the Department of Computer Science at Victoria University of Wellington, New Zealand. His main research interest lies in examining how programming language features affect the production of code. His current focus is on the impact that programming languages, particularly object-oriented languages, have on the reusability of the resulting code. He is also interested in various aspects of distributed systems.

Gavin Turner received a Ph.D. in Computer Science at the University of Newcastle, Australia in 1996, and 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.

Optimal Dimension-Exchange Token Distribution on Complete Binary Trees

Michael E. Houle

Department of Computer Science and Software Engineering, University of Newcastle, Callaghan NSW, Australia

Ewan Tempero

Department of Computer Science, Victoria University of Wellington, Wellington, New Zealand

Gavin Turner

Department of Computer Science, Victoria University of Wellington, Wellington, New Zealand

Abstract

Load balancing on a multi-processor system involves shifting work around the system so that each processor has about the same amount of processing to perform. The *token-distribution* problem is a static variant of the load balancing problem for the case in which the workloads in the system cannot be divided arbitrarily; that is, where each token represents an atomic element of work. A simple, scalable method for distributing tokens over a distributed-memory parallel architecture is the so-called *dimension-exchange* approach, which is based on the repetitive application of an extremely simple and scalable local exchange protocol. The behaviour of this approach depends on the topology of the interconnection network.

This paper presents the analysis of dimension-exchange algorithms for token distribution on the complete binary tree. We show that for the height H complete binary tree and any initial distribution for which the discrepancy in workloads is greater than H tokens, the dimension-exchange approach leads to the eventual convergence of the distribution such that the discrepancy is at most H. Furthermore, we show that the rate of this convergence is optimal with respect to the bisection width lower bound. These results are the first to establish that dimension-exchange techniques lead to optimal algorithms for finitely-divisible load balancing on a tree-connected network.

Preprint submitted to Elsevier Science *30 May 1997*

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. 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 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.

There are many data distribution methods that achieve a balanced token distribution by gathering and making use of a certain amount of global information [DG94,MadHOW93,TS95]. Such methods are often unsatisfactory, in that they do not take into account the practical limitations of the parallel architecture, or result in algorithms that are unnecessarily complex. One method that requires no such global information is the so-called *dimension-exchange* method, which is based on the repetitive application of an extremely simple and scalable local exchange protocol. To be able to implement a dimension-exchange algorithm on a particular parallel architecture, the communication edges of the underlying topology must be *partitionable* (or *colourable*) into sets whereby no two edges of the same set are incident on the same processor. For networks having hypercube or mesh-connected topologies, the edges can be partitioned in a natural fashion, according to the *dimension* of the network along which the edge is oriented. For other networks, partitions may be based on sets of *matchings* [HLM$^+$90].

Dimension-exchange algorithms use the edge-colouring of a network to pair processors for data exchange, and are invariably of the following general form:

Dimension-Exchange Algorithm
LOOP
  FOR i = colour 1 to κ (*κ *colours* *)
    Over all pairs of PEs connected by edges
    of colour i, compare and exchange values;
END

Due to their simplicity and scalability, many researchers have studied the applicability of dimension-exchange techniques to load balancing problems; the first being Cybenko [Cyb89] in 1987, who proposed an algorithm for the d-dimensional hypercube under the assumption that the load in each PE was *infinitely-divisible* — that is, a real-valued quantity able to be split among processors in an arbitrary fashion. Cybenko showed that if every exchange results in an equal sharing of the load between the two PEs involved, then after d iterations the PE loads would be perfectly balanced.

This original work prompted a steady stream of research into the analysis of dimension-exchange algorithms. In 1990, Hosseini *et al.* demonstrated that, for infinitely-divisible loads, Cybenko's analysis could be generalised to arbitrary k-colourable networks [HLM$^+$90]. In 1992 Xu and Lau [XL92a] extended the work of Hosseini *et al.* by showing that for some topologies, the rate at which the global discrepancy converged to zero could be optimised by altering the ratio with which infinitely-divisible loads were locally balanced. Later that year the same authors [XL92b] computed optimal ratios for the linear array, ring, 2-dimensional mesh and 2-dimensional torus.

To date, a large body of results exist detailing the performance of the dimension-exchange approach over infinitely-divisible loads; on the other hand, little has been known concerning dimension-exchange under the more realistic assumption of *finitely-divisible* loads, that is, loads representable as a set of tokens.

In 1988 Ranka, Won, and Sahni [RWS88] studied the operation of Cybenko's algorithm empirically for the d-dimensional hypercube assuming finitely-divisible loads. They observed that the difference between the maximum number and minimum number of tasks over all PEs of the network (called the *discrepancy*) would eventually fall to at most d. Hosseini *et al.* [HLM$^+$90] and Plaxton [Pla89] independently confirmed this observation by providing algorithms which, after d steps, reduced the discrepancy to at most d.

In 1996, Houle and Turner [HT96] proposed and analysed an optimal dimension-exchange algorithm for the two dimensional mesh and torus. The algorithm was shown to reduce the discrepancy to a level equal to the minimum degree of the mesh/torus in time optimal with respect to the bisection width. Their analysis was the first to show that dimension-exchange algorithms could optimally distribute data over architectures of constant degree.

This paper presents the analysis of a dimension-exchange algorithm for token distribution on the complete binary tree. We show that, under the action of the algorithm, the global discrepency converges to the height of the tree, and the rate of this convergence is optimal with respect to the bisection width of the architecture. In an earlier version of this paper, [TT97], we managed to prove the convergence result, but were unable to establish the rate of convergence. In this paper we present a new proof which, as well as identifying the limit of convergence, conveniently provides us with the rate.

The organisation of the paper is as follows: in the next section, we describe the model of computation. In Section 2, we propose a dimension-exchange algorithm for the complete binary tree. The notation and preliminary concepts that we use in the analysis of the algorithm are introduced in Section 3. The analysis of the convergence properties of the algorithm appears in Section 4. Finally, concluding remarks are made in Section 5.

2 Token Distribution

2.1 Problem

The token distribution problem was first posed by Peleg and Upfal ([PU87]) and may be stated as follows.

Definition 1 *Let $\mathcal{A}$ be a parallel architecture with N processors $P_1, \ldots, P_N$, each processor P_i, $1 \leq i \leq N$, containing a stack of $\mu \leq l_i \leq M$ tokens, where $\Delta = M - \mu$. Then the* token distribution problem, *denoted,* $\mathrm{TD}(\mathcal{A}; \Delta, M, \delta)$, *is the problem of distributing the tokens so that each processor P_i ends up with a stack of $\mu' \leq l'_i \leq M'$ tokens, where $\delta = M' - \mu'$.*

The difference between the smallest and largest stacks in the system, Δ, is the *global discrepancy.* Therefore, $TD(A; \Delta, M, \delta)$ may be reinterpreted as asking whether the global discrepancy can be reduced from Δ to δ.

2.2 Model of Computation

We consider a parallel architecture in which the processors are connected via an interconnection network based on the *complete binary tree* [Lei92]. A complete binary tree of height H, $\mathcal{T}_H$, is a binary tree in which every node has either zero or two children, and all leaves are at distance H from the root. The breadth-first numbering for such a tree numbers the root with 0, and for every node i, $0 \leq i < 2^{H+1} - 1$, its left subchild is numbered $2 \cdot i + 1$, and its right subchild is numbered $2 \cdot i + 2$. For every node i, $0 < i < 2^{H+1} - 1$, its parent, denoted $\mathsf{parent}(i)$, is numbered $\lceil \frac{i}{2} \rceil - 1$. Node i is said to exist at depth $\lfloor \log_2 (i+1) \rfloor$. Figure 1 shows a complete binary tree of height 3.

We assume that each PE has facilities for one-port communication. Under this assumption, the PEs are connected to their neighbours by uni-directional communication links, and may send or receive at most one message at any one time. This model is considerably weaker than the MIMD-model, where bi-directional links are assumed and concurrent communication to all the neighbours is allowed.

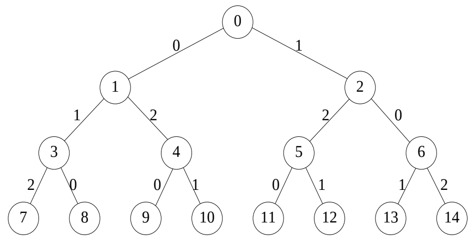

Fig. 1. Depth-First colouring of edges of the height 3 complete binary tree.

In this paper, we show that the dimension-exchange technique may be used to solve $TD(\mathcal{T}_H; \Delta, M, H)$ for $\Delta \geq H$; that is, under the algorithm we propose, the global discrepancy Δ in PE loads converges to H.

2.3 Dimension-Exchange Algorithm

To implement any dimension-exchange algorithm, two choices must be made: the *network colouring* over which the algorithm will be applied, and a suitable *comparison-exchange step.*

The algorithm presented in this paper, TreeBalance, is based on the so-called *Depth-First* colouring illustrated in Figure 1, in which the connections are consecutively coloured (mod 3) based on a depth-first traversal of the binary tree. All edges are coloured via the following recursive rule, where ColourTree($\mathcal{T}_H$,0) begins the process.

```
ColourTree(Root, c);
BEGIN
   Root.left_child_edge.Colour = c
   Root.right_child_edge.Colour = (c+1) mod 3
   ColourTree(Root.left_subtree, (c+1) mod 3)
   ColourTree(Root.right_subtree, (c+2) mod 3)
END
```

The local comparison-exchange step used by the algorithm can be informally stated as follows:

[Plus-Minus1]: Neighbouring PEs compare their loads and, if the loads differ, a *single token* is sent from the heavily-loaded PE to the lightly-loaded PE.

Depending on the architecture, implementing such an elementary operation may require several (ie. a constant number of) communication steps. A formal definition will be given in the next section.

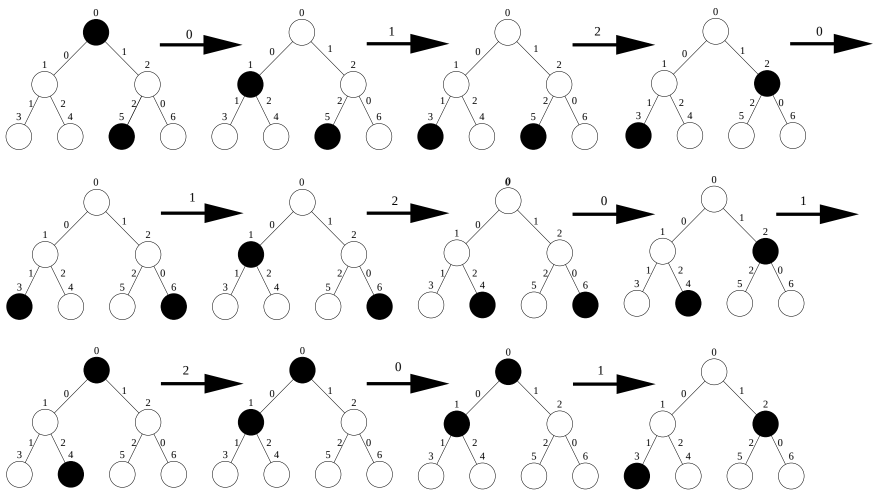

Fig. 2. Algorithm TreeBalance operating on a tree of height 2.

Based on this elementary operation, algorithm TreeBalance is given below:

```
TreeBalance
BEGIN
    i = 0
    REPEAT
    BEGIN
        Apply elementary step [Plus-Minus1]
            over all network connections having
            colour i mod 3
        i++
    ENDREPEAT
END
```

For our analysis, it is convenient to assume that one iteration of the loop in this algorithm takes place in one unit of time.

In the remainder of the paper, we shall prove that under TreeBalance, any initial distribution on the complete binary tree of height H for which $\Delta > H$ converges to a distribution for which $\Delta \leq H$.

3 Notation and Preliminaries

In this section, we present notation and properties needed to analyse the convergence properties of TreeBalance over $\mathcal{T}_H$.

The operation of the algorithm over a complete binary tree of height 2 is illustrated in Figure 2, and can be described as follows. Consider the situation where the root node of tree $\mathcal{T}_H$ initially contains one token, as does the first leaf of the second subtree (node 5 in Figure 2), and all other nodes contain zero tokens. The algorithm begins by comparing and exchanging across edges of colour 0. In the example, the only such edge that has an imbalanced load is that between nodes 0 and 1 (there is no 0 coloured edge incident on node 5). So, on the first step, node 0 will reduce its load by 1 while node 1 will increase its load, resulting in the token *moving from node* 0 *to node* 1. On the second step (edges coloured 1), the token will move to node 3. On the third step, the token at node 3 has nowhere to go, since there is no edge coloured 2 incident on that node, however now the token at node 5 can move to node 2.

The token from the root will continue to move, traversing the tree in a depth-first fashion, until it has visited all nodes of the tree (at which point it arrives back at the root). Furthermore, each node in the tree (including the root) will have been visited exactly 3 times by the token. Note that at the root and the leaves, tokens are delayed 1 and 2 steps respectively. In this context, each step that a token is delayed at a node is identified as another visit to that node. We denote such a path as a *cycle* or *traversal* of the tree, representing a path of exactly $3 \cdot N$ ($N = 2^{H+1} - 1$) steps. In the sequel, we will frequently refer to a traversal of a subtree. Unlike the full tree, a traversal of a subtree is not a cycle. However the root of the subtree will still be visited exactly 3 times: once on entering the subtree, once when moving from the subtree's left child to its right child, and finally just before the traversal leaves the subtree. Thus, the only difference between a traversal of a subtree and a traversal of the tree itself is the 1 step delay at the root. Consequently we have:

Fact 2 *If $\mathcal{R}$ is a subtree of $\mathcal{T}_H$ of height h, then a complete traversal of $\mathcal{R}$ takes $3 \cdot (2^{h+1} - 1) - 1$ steps, that is, if an object is at the root of $\mathcal{R}$ at time t, then it will next be at the parent of the root of $\mathcal{R}$ at time $t + 3 \cdot (2^{h+1} - 1)$.*

Like its counterpart, the token that begins at node 5 will traverse the tree, using exactly the same path taken by the root token, but *exactly 12 steps ahead* of it. This means that the token originating at the root will arrive at a particular node on a particular edge exactly 12 steps *after* the token originating from node 5. Conversely, this token may also be considered *exactly 9 steps behind* the root token (since the traversal is a cycle). Interestingly, the 10th step of the algorithm sees the current node containing the token from the root, being compared to the node containing the token from node 5. Now the outcome of this comparison is such that the loads of two nodes remain the same - no tokens are exchanged. However, if the two tokens of interest are to finish their traversal, it is necessary for us to imagine that they have *swapped position* - in essence we imagine that the tokens have moved through each other!

3.2 Cycle Traversal

In general, two such tokens that begin at different nodes remain a constant distance apart with respect to the traversal. This observation is more obvious if we represent the complete tree traversal as a cyclic graph with $3 \cdot (2^{H+1} - 1)$ edges and $3 \cdot (2^{H+1} - 1)$ nodes. Such a graph is shown in Figure 3 for a tree of height 3. Note that each node appears exactly 3 times in the traversal, each edge appears exactly twice, and dummy edges exist corresponding to delays at the leaves and root. The positions at which a token may start (those which are incident on colour 0) have been darkly shaded. Note that the distance between tokens is always a positive multiple of 3. This is because a token arrives at a particular node on a particular edge. Since this edge has some colour c, tokens may only arrive at this node, on this edge, every 3rd step.

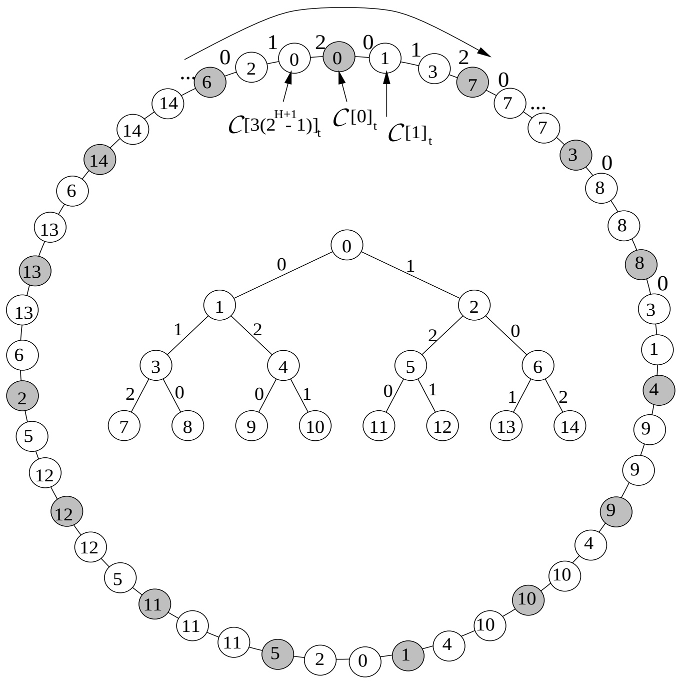

Fig. 3. Traversal illustrated for a Complete Binary Tree of Height 3.

We may generalise this idea of cyclic traversal for single tokens to groups or *piles* of tokens. That is, instead of viewing the positions of tokens as being tied to *absolute* (fixed) locations of $\mathcal{T}_H$, it will often be convenient to view their positions *relative* to a position in a *traversal* of the tree. Under the relative interpretation of algorithm TreeBalance, we imagine a set of N *piles* of tokens, (each a group of zero or more), which circulate around the tree in a manner

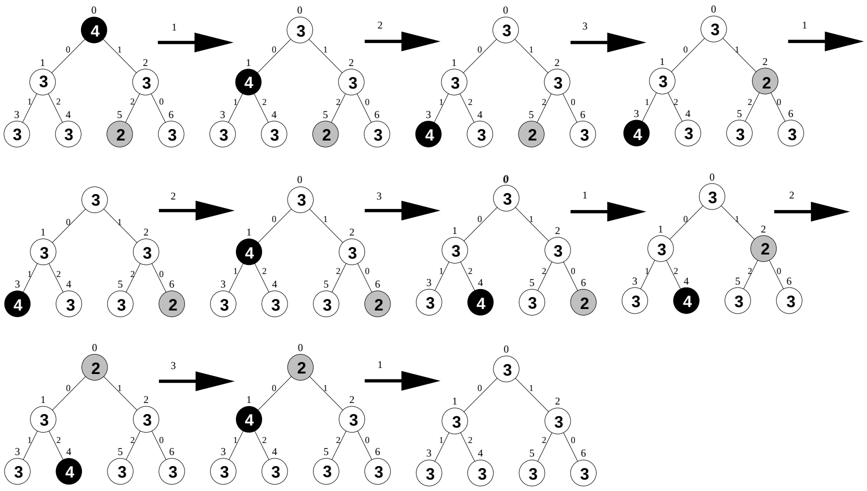

Fig. 4. A pile of height 2 (Lightly Shaded) and of cyclic distance 12 ahead of another pile of height 4 (Darkly Shaded) collides with the second pile and their heights are altered.

similar to the single tokens we tracked in the preceding section. At any time instant each node i, $0 \le i < N$, of tree $\mathcal{T}_H$ is the *location* of a particular pile α, denoted $i = \mathcal{L}[\alpha]_t$. At each time step, the pile moves to the next node in the cyclic traversal, and the size of the pile changes depending on the outcome of the associated comparison-exchange operation.

Although piles remain a constant distance apart with respect to the traversal, they will still (periodically) come into contact due to the topology of the tree. Figure 4 shows an example of this.

3.3 Notation

The number of tokens in pile α after t steps of TreeBalance will be called its *size*, and will be denoted by $\mathcal{S}[\alpha]_t$. The number of tokens at node j at time t (that is, under the absolute interpretation) is the node's *height*, denoted $\mathcal{H}[j]_t$.

Initially we have, for any pile α, $\mathcal{S}[\alpha]_0 = \mathcal{H}[\mathcal{L}[\alpha]_0]_0$. If a pile has size that is minimum or maximum over all positions of $\mathcal{T}_H$, we simply refer to it as a *minimum* or *maximum* respectively. For any subtree $\mathcal{R}$ of $\mathcal{T}_H$, let $\mathsf{min}[\mathcal{R}]_t$ be the size of the minimum pile in $\mathcal{R}$ at time t, and let $\mathsf{max}[\mathcal{R}]_t$ the the size of the maximum pile in $\mathcal{R}$ at time t.

We can now give a formal definition for the comparison-exchange step:

Definition 3 ([Plus-Minus1] Absolute) *If two nodes i and j are compared at time t, then*

if $\mathcal{H}[i]_{t-1} > \mathcal{H}[j]_{t-1}$ *then*
  $\mathcal{H}[i]_t = \mathcal{H}[i]_{t-1} - 1$
  $\mathcal{H}[j]_t = \mathcal{H}[j]_{t-1} + 1$
else if $\mathcal{H}[i]_{t-1} < \mathcal{H}[j]_{t-1}$ *then*
  $\mathcal{H}[i]_t = \mathcal{H}[i]_{t-1} + 1$
  $\mathcal{H}[j]_t = \mathcal{H}[j]_{t-1} - 1$
else
  $\mathcal{H}[i]_t = \mathcal{H}[i]_{t-1}$
  $\mathcal{H}[j]_t = \mathcal{H}[j]_{t-1}$
endif

With respect to relative locations, the comparison-exchange rule of TreeBalance can be reinterpreted. Consider nodes i and j joined by colour c at time t. If $t \bmod 3 = c$, then the loads of nodes i and j are compared, tokens are possibly exchanged, and the piles α and β swap locations, that is

Definition 4 *If two piles α and β are compared at time t, then their positions are swapped; that is*

(i) $\mathcal{L}[\beta]_t = \mathcal{L}[\alpha]_{t-1}$
(ii) $\mathcal{L}[\alpha]_t = \mathcal{L}[\beta]_{t-1}$

We can now reinterpret the comparison-exchange operation under the relative interpretation:

Property 1 ([Plus-Minus1] Relative) *If two piles α and β are compared at time t, then*

if $\mathcal{S}[\alpha]_{t-1} > \mathcal{S}[\beta]_{t-1} + 1$ *then*
  $\mathcal{S}[\alpha]_t = \mathcal{S}[\beta]_{t-1} + 1$
  $\mathcal{S}[\beta]_t = \mathcal{S}[\alpha]_{t-1} - 1$
else if $\mathcal{S}[\alpha]_{t-1} + 1 < \mathcal{S}[\beta]_{t-1}$ *then*
  $\mathcal{S}[\alpha]_t = \mathcal{S}[\beta]_{t-1} - 1$
  $\mathcal{S}[\beta]_t = \mathcal{S}[\alpha]_{t-1} + 1$
else
  $\mathcal{S}[\beta]_t = \mathcal{S}[\alpha]_{t-1}$
  $\mathcal{S}[\alpha]_t = \mathcal{S}[\beta]_{t-1}$
endif

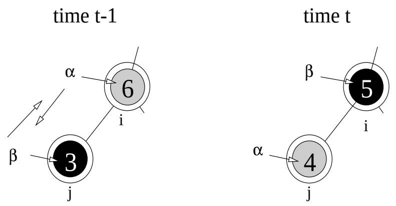

Fig. 5. Comparison Exchange operation between neighbouring nodes leads to the relocation and the size alteration of respective piles.

Proof. Let i and j be nodes such that $i = \mathcal{L}[\alpha]_{t-1}$ and $j = \mathcal{L}[\beta]_{t-1}$. Consider the case where $\mathcal{S}[\alpha]_{t-1} > \mathcal{S}[\beta]_{t-1} + 1$. Then $\mathcal{H}[i]_{t-1} > \mathcal{H}[j]_{t-1} + 1$ and so by definition 3, $\mathcal{H}[i]_t = \mathcal{H}[i]_{t-1} - 1$. By definition 4, $\mathcal{L}[\beta]_t = \mathcal{L}[\alpha]_{t-1} = i$ and so $\mathcal{S}[\beta]_t = \mathcal{H}[i]_t = \mathcal{H}[i]_{t-1} - 1 = \mathcal{S}[\alpha]_{t-1} - 1$, as required. The arguments for the other cases are similar. □

An example of this behaviour is illustrated in Figure 5.

The following two properties relating to the behaviour of the size of piles are important for our analysis. Informally, the first property states that if the sizes of two piles are at least as large (small) as some value B when they are compared, then their resulting sizes will still be at least as large (small) as B after the comparison. The interpretation is that all pile sizes move towards the average.

Property 2 *Consider any two piles, α at location i and β at location j that are compared at time t, and some integer B:*

(i) If $\mathcal{S}[\alpha]_{t-1}, \mathcal{S}[\beta]_{t-1} \geq B$, then $\mathcal{S}[\alpha]_t, \mathcal{S}[\beta]_t \geq B$, and
(ii) If $\mathcal{S}[\alpha]_{t-1}, \mathcal{S}[\beta]_{t-1} \leq B$, then $\mathcal{S}[\alpha]_t, \mathcal{S}[\beta]_t \leq B$.

Proof.

(i) Consider what happens to the size of α (the argument for β is similar). There are three possibilities:
   (a) $\mathcal{S}[\alpha]_t > \mathcal{S}[\beta]_{t-1}+1$: By hypothesis, $\mathcal{S}[\beta]_{t-1} \geq B$ so $\mathcal{S}[\alpha]_t > B+1 > B$
   (b) $\mathcal{S}[\alpha]_{t-1}+1 < \mathcal{S}[\beta]_{t-1}$: By hypothesis $B \leq \mathcal{S}[\alpha]_{t-1}$, so $\mathcal{S}[\beta]_{t-1} \geq B+1$ and so $\mathcal{S}[\alpha]_t = \mathcal{S}[\beta]_{t-1} - 1 > B - 1 \geq B$
   (c) $\mathcal{S}[\alpha]_t = \mathcal{S}[\beta]_{t-1}$: This follows directly from the hypothesis.
(ii) The argument is similar. □

The second property, being a direct implication of property 2, states that the minimum (or maximum) for $\mathcal{T}_H$ can only increase (or decrease).

Property 3 *For all times t and $t' > t$, $\mathsf{min}[\mathcal{T}_H]_t \leq \mathsf{min}[\mathcal{T}_H]_{t'}$ and $\mathsf{max}[\mathcal{T}_H]_t \geq \mathsf{max}[\mathcal{T}_H]_{t'}$.*

Proof. Consider the situation at time t. Assume, by way of contradiction, that there is a time $t' > t$ such that $\mathsf{min}[\mathcal{T}_H]_{t'} < \mathsf{min}[\mathcal{T}_H]_t$. We can assume without loss of generality that t' is the smallest time that satisfies this condition.

Let α be the pile that becomes the minima, that is, $\mathcal{S}[\alpha]_{t'} = \mathsf{min}[\mathcal{T}_H]_{t'}$. At time $t'-1$, it must be the case that $\mathcal{S}[\alpha]_{t'-1} \geq \mathsf{min}[\mathcal{T}_H]_t$, since otherwise t' is not the smallest to satisfy $\mathsf{min}[\mathcal{T}_H]_{t'} < \mathsf{min}[\mathcal{T}_H]_t$. Let β be the pile that α is compared with at time t'. Similarly then, it must be the case that $\mathcal{S}[\beta]_{t'-1} \geq \mathsf{min}[\mathcal{T}_H]_t$. By property 2, it must be that $\mathcal{S}[\alpha]_{t'}, \mathcal{S}[\beta]_{t'} \geq \mathsf{min}[\mathcal{T}_H]_t$, and so we have the required contradiction.

The argument for the maximum is similar. $\square$

4 Analysis of TreeBalance

In this section, we show that algorithm TreeBalance for complete binary trees of height H converges in time linear to the reduction in global discrepancy. The convergence of Algorithm TreeBalance is stated and proven in theorem 10 at the end of this section. To prove this theorem we present several supporting lemmas. Before we do, let us consider a lower bound on the rate of convergence. For a complete binary tree $\mathcal{T}_H$ of height H, a lower bound on the time required to solve token distribution problem $TD(\mathcal{T}_H; \Delta, M, \delta)$ is easily derived. The result is stated in Theorem 5.

Theorem 5 (Lower Bound) *Let $\mathcal{T}_H$ be a complete binary tree of height H, and $N = 2^{H+1} - 1$ nodes. Any solution to problem $\mathrm{TD}(\mathcal{T}_H; \Delta, M, \delta)$ 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 $(N-1)/2$ PEs in the left subtree of $\mathcal{T}_H$ contain M tokens, and the remaining $(N-1)/2$ PEs in the right subtree contain μ tokens. To reduce Δ to δ, $\frac{\Delta-\delta}{2} \cdot \frac{(N-1)}{2}$ tokens must be moved from the PEs with M tokens to the PEs with μ tokens. Given that the bisection width of $\mathcal{T}_H$ is $\mathcal{O}(1)$, this requires at least $\Omega((\Delta-\delta) \cdot N)$ steps.

Lemma 6 is the first of the supporting lemmas for theorem 10. Informally, this lemma establishes that the only way the minima for a subtree can be reduced is if, at the time the root of the subtree is compared with its parent, the pile at the root is a minima for the subtree.

Lemma 6 *Consider a complete subtree $\mathcal{R}$ rooted at node r at time t. Then the minimum for $\mathcal{R}$ will decrease at time $t' > t$, that is, $\mathsf{min}[\mathcal{R}]_{t'-1} = \mathsf{min}[\mathcal{R}]_t$ and $\mathsf{min}[\mathcal{R}]_{t'} = \mathsf{min}[\mathcal{R}]_t - 1$,* if and only if *at time $t' - 1$,*

(i) $\mathcal{H}[r]_{t'-1} = \mathsf{min}[\mathcal{R}]_t$ (there is a minima at the root of $\mathcal{R}$),
(ii) $\mathcal{H}[\mathsf{parent}(r)]_{t'-1} < \mathsf{min}[\mathcal{R}]_t$ (r's parent has a smaller load), *and*
(iii) at time t', r is compared with its parent.

Proof.

if: Assume that at time t', $\mathcal{H}[r]_{t'-1} = \mathsf{min}[\mathcal{R}]_t$, $\mathcal{H}[\mathsf{parent}(r)]_{t'-1} < \mathsf{min}[\mathcal{R}]_t$, and at time t' r is compared with its parent.

Since $\mathcal{H}[\mathsf{parent}(r)]_{t'-1} < \mathsf{min}[\mathcal{R}]_t$, by definition 3, $\mathcal{H}[r]_{t'} = \mathcal{H}[i]_{t'-1} - 1$. By the definition of $\mathsf{min}[\mathcal{R}]_{t'}$, $\mathsf{min}[\mathcal{R}]_{t'} \leq \mathcal{H}[r]_{t'} = \mathcal{H}[i]_{t'-1} - 1 = \mathsf{min}[\mathcal{R}]_t - 1$.

Now consider any nodes $s_1, s_2 \neq r$ in $\mathcal{R}$. At time $t'-1$, by the definition of $\mathsf{min}[\mathcal{R}]_{t'-1}$, $\mathcal{H}[s_1]_{t'-1}, \mathcal{H}[s_2]_{t'-1} \geq \mathsf{min}[\mathcal{R}]_{t'-1}$, and so by property 2, it must be the case that $\mathcal{H}[s_1]_{t'}, \mathcal{H}[s_2]_{t'} \geq \mathsf{min}[\mathcal{R}]_{t'-1}$. That is, no other piles can contribute to the minimum of $\mathcal{R}$ and hence, $\mathsf{min}[\mathcal{R}]_{t'} = \mathsf{min}[\mathcal{R}]_t - 1$.

only if: Now assume $\mathsf{min}[\mathcal{R}]_{t'-1} = \mathsf{min}[\mathcal{R}]_t$ and $\mathsf{min}[\mathcal{R}]_{t'} < \mathsf{min}[\mathcal{R}]_t$. Consider any two piles, α and β in $\mathcal{R}$ and being compared at time $t'-1$. Then by the hypothesis, $\mathcal{S}[\alpha]_{t'-1} \geq \mathsf{min}[\mathcal{R}]_{t'-1} = \mathsf{min}[\mathcal{R}]_t$ and $\mathcal{S}[\alpha]_{t'-1} \geq \mathsf{min}[\mathcal{R}]_{t'-1} = \mathsf{min}[\mathcal{R}]_t$. Hence by property 2, $\mathcal{S}[\alpha]_{t'}, \mathcal{S}[\beta]_{t'} \geq \mathsf{min}[\mathcal{R}]_t$. Thus the only piles that can end up with size less than $\mathsf{min}[\mathcal{R}]_t$ at time t' are those with at least one of them *not* in $\mathcal{R}$. If both piles are not in $\mathcal{R}$ then $\mathsf{min}[\mathcal{R}]_{t'}$ cannot change, so one pile must be in $\mathcal{R}$ and the other is not and the only way that can happen is if one pile is at r and the other is at $\mathsf{parent}(r)$.

Since the only comparison taking place at time t' that affects the minimum of $\mathcal{R}$ is between r and $\mathsf{parent}(r)$, if $\mathsf{min}[\mathcal{R}]_{t'} < \mathsf{min}[\mathcal{R}]_{t'-1}$ then the pile at r must have reduced in height (reducing the height of $\mathsf{parent}(r)$ does not affect $\mathsf{min}[\mathcal{R}]_{t'}$ since $\mathsf{parent}(r)$ is not in $\mathcal{R}$.). Thus $\mathcal{H}[r]_{t'-1} = \mathsf{min}[\mathcal{R}]_t$. The only way the pile at r can be reduced in height is, by definition 3, if $\mathcal{H}[\mathsf{parent}(r)]_{t'-1} < \mathcal{H}[r]_{t'-1}$. □

Lemma 7 establishes that if there is one minima in a subtree of height h, then that minima has to traverse the entire subtree, which takes $3 \cdot (2^{h+1} - 1) - 1$ steps, before there is any possibility that pile can become smaller.

Lemma 7 *Let $\mathcal{R}$ be a subtree of $\mathcal{T}_H$ of height h rooted at node r. Assume that at time t there is exactly one minima in $\mathcal{R}$ and that pile of size $\mathsf{min}[\mathcal{R}]_t$ is at r. If at the next step of the algorithm this pile will be compared with the root of $\mathcal{R}$ś left subtree, then $\forall t'$, $t \le t' < t + 3 \cdot (2^{h+1} - 1)$, $\mathsf{min}[\mathcal{R}]_{t'} \ge \mathsf{min}[\mathcal{R}]_t$.*

Proof. By lemma 6, the only way the minimum pile height can get smaller is if the pile at r, when r is compared with its parent, is a minima. The hypothesis means that the minima of size $\mathsf{min}[\mathcal{R}]_t$ must move to the left child of the root on the next step. (Since it is the sole minima, it must be strictly smaller than the left child pile. However note that it may change size in doing so.) By fact 2, it will not return to $\mathsf{parent}(r)$ until time $t + 3 \cdot (2^{h+1} - 1)$. Since the sole minima for $\mathcal{R}$ will not be at r for time t', $t \le t' < t + 3 \cdot (2^{h+1} - 1)$, part 1 of lemma 6 (only if) will not be satisfied until at least time $t + 3 \cdot (2^{h+1} - 1)$, and so the result follows. □

Lemma 8 establishes that the minima for a subtree will not decrease by more than 1 during the time it takes for *any* pile to traverse the subtree.

Lemma 8 *Let $\mathcal{R}$ be a subtree of $\mathcal{T}_H$ of height h rooted at node r. Then, for any time t and any time t' such that $t \le t' < t + 3 \cdot (2^{h+1} - 1)$, $\mathsf{min}[\mathcal{R}]_{t'} \ge \mathsf{min}[\mathcal{R}]_t - 1$*

Proof. If at time t, the pile at the root of the subtree is the sole minima for the subtree and it has just entered the subtree, then by lemma 7, the result follows in this case. Otherwise, let t' be the smallest time such that $t' > t$ and $\mathsf{min}[\mathcal{R}]_{t'} < \mathsf{min}[\mathcal{R}]_t$. Then there are two cases:

(i) $t' \ge t + 3 \cdot (2^{h+1} - 1)$, in which case there is nothing to prove, or
(ii) $t' < t + 3 \cdot (2^{h+1} - 1)$. In this case, the three "only if" conditions of lemma 6 must hold, in particular, $\mathsf{min}[\mathcal{R}]_{t'} = \mathsf{min}[\mathcal{R}]_t - 1$. By lemma 7, the size of the minima cannot be reduced again until time $t' + 3 \cdot (2^{h+1} - 1) + 1 > t + 3 \cdot (2^{h+1} - 1)$, which proves the result. □

We are now able to present the main theorem from which we will derive the existence and rate of convergence results. Informally, the theorem asserts that the only way a maximum pile can not be reduced during the traversal of a subtree is if all the minima in the subtree are not too small.

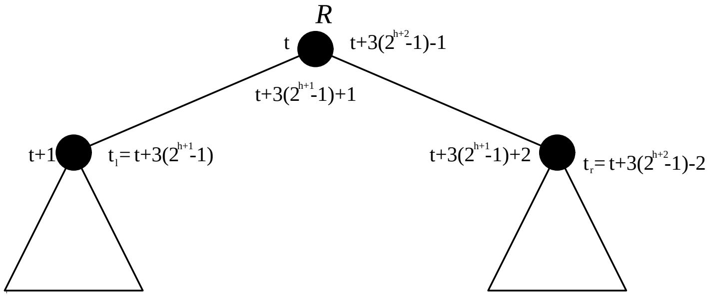

Fig. 6. Movement of a pile around a subtree of height $h+1$.

Theorem 9 *Let $\mathcal{R}$ be a subtree of $\mathcal{T}_H$ of height h rooted at node r and consider a time t such that $\mathcal{H}[r]_t = \mathsf{max}[\mathcal{R}]_t$ and at the next step, r will be compared with its left child (that is, a maxima for $\mathcal{R}$ has just entered the subtree). Let α be the pile at r ($\mathcal{L}[\alpha]_t = r$). If, for all time t', $t \leq t' < t + 3 \cdot (2^{h+1} - 1)$, $\mathcal{S}[\alpha]_{t'} = \mathsf{max}[\mathcal{R}]_t (= \mathcal{S}[\alpha]_t)$ then $\mathsf{min}[\mathcal{R}]_{t'} \geq \mathsf{max}[\mathcal{R}]_t - h$.*

Proof. The proof is by induction on the height h of $\mathcal{R}$.

Basis: Consider a subtree $\mathcal{R}$ of height $h = 0$ (that is, a leaf). The maximum pile does not leave the leaf until time $t' = t + 3 = t + 3 \cdot (2^{h+1} - 1)$, and before that is not compared with anything (and so cannot decrease in size).

Induction hypothesis: Assume that the theorem is true for subtrees of height h.

Induction step: Consider a subtree $\mathcal{R}$ of height $h+1$.

At time t, α is at node r, having come from $\mathsf{parent}(r)$. At time $t+1$, α will be at the root of $\mathcal{R}$ś left subtree ($\mathcal{R}_l$, of height h). By fact 2, after another $3 \cdot (2^{h+1} - 1) - 1$ steps, that is at time $t_l = t + 3 \cdot (2^{h+1} - 1)$ steps, α will have traversed $\mathcal{R}_l$ and will be back at the root of $\mathcal{R}_l$.

At time $t_l + 1$, α will be at r and at time $t_l + 2$, α will have moved to the root of $\mathcal{R}$'s right subtree ($\mathcal{R}_r$, of height h), beginning a traversal of that subtree. Again by fact 2, after another $3 \cdot (2^{h+1} - 1) - 1$ steps, that is at time $t_r = t_l + 2 + 3 \cdot (2^{h+1} - 1) - 1 = t + 2 \cdot 3 \cdot (2^{h+1} - 1) + 1 = t + 3 \cdot (2^{h+2} - 1) - 2$, the maximum will have finished this traversal and have returned to the root of $\mathcal{R}_r$. At time $t' = t_r + 1 = t + 3 \cdot (2^{h+2} - 1) - 1$ the maximum will reach r ready to leave $\mathcal{R}$. This is illustrated in figure 6

By the induction hypothesis, at time t_l, $\mathsf{min}[\mathcal{R}_l]_{t_l} \geq \mathsf{max}[\mathcal{R}_l]_{t+1} - h$. The hypothesis assumes that α does not change size, and so by property 3, $\mathsf{max}[\mathcal{R}_l]_{t+1} = \mathsf{max}[\mathcal{R}]_t$, and so we have $\mathsf{min}[\mathcal{R}_l]_{t_l} \geq \mathsf{max}[\mathcal{R}_l]_t - h$.

It must be that the steps at time $t_l + 1$ and $t_l + 2$ cannot alter the minima for the left subtree (that is, $\mathsf{min}[\mathcal{R}_l]_{t_{l+2}} = \mathsf{min}[\mathcal{R}_l]_{t_l}$) since at time $t_l + 1$ α is compared with the pile at the root, and by hypothesis α does not change height, and at time $t_l + 2$ the root node of the left subtree is not compared with r (so the "only if" for lemma 6 is not satisfied). Lemma 8 gives us that the size of the minimum pile in the left subtree cannot decrease by more than one while α traverses $\mathcal{R}_r$, that is, for any time t'', $t_l + 2 \leq t'' < t_l + 2 + 3 \cdot (2^{h+1} - 1) - 1 = t_r + 1$, $\mathsf{min}[\mathcal{R}_l]_{t''} \geq \mathsf{min}[\mathcal{R}_l]_{t_l} - 1 \geq \mathsf{max}[\mathcal{R}]_t - h - 1$.

At this point, we have shown the result only for α reaching the root of $\mathcal{R}_r$. We now have to show that the minimum for $\mathcal{R}$ does not change as α moves from the root of $\mathcal{R}_r$ to r. On this step, all the comparisons that take place occur either in the subtrees, and so by property 2 cannot reduce the minimum, or between α and the pile at r at time t_r. By hypothesis, α does not change size, and so it must be the case that $\mathcal{H}[r]_{t_r} \geq \mathcal{S}[\alpha]_{t_r} - 1 = \mathsf{max}[\mathcal{R}]_t - 1$, and so the minimum cannot be affected.

At time $t_r + 1$, α will be at r, and, by the argument above, $\mathsf{min}[\mathcal{R}]_{t_r+1} \geq \mathsf{max}[\mathcal{R}]_t - h - 1 = \mathsf{max}[\mathcal{R}]_t - (h+1)$, and therefore the theorem holds. □

We can now show that $\mathsf{TreeBalance}$ will reduce the global discrepancy Δ to H in time no greater than $2 \cdot 3 \cdot N \cdot (\Delta - H)$.

Theorem 10 (Convergence) $\mathsf{TreeBalance}$ *will solve* $\mathrm{TD}(\mathcal{T}_H; \Delta, M, \delta)$ *for* $\Delta > H$, $\Delta > \delta \geq H$, *in* $2 \cdot 3 \cdot N \cdot (\Delta - \delta)$ *steps.*

Proof. First note that it takes a pile $3 \cdot N = 3 \cdot (2^{H+1} - 1)$ steps to completely traverse $\mathcal{T}_H$. We will show that when $\Delta > H$ the global discrepancy is $\Delta - 1$ after $2 \cdot 3 \cdot N$ steps.

Assume that the pile at the root of $\mathcal{T}_H$ is a maximum, that is, has height $\mathsf{max}[\mathcal{T}_H]_0$. Then after $3 \cdot N$ steps it must be the case that the height of this pile has changed, since otherwise, by theorem 9, the height of all other piles in $\mathcal{T}_H$ are at least $\mathsf{max}[\mathcal{T}_H]_0 - H$, that is, $\Delta = H$, which contradicts the hypothesis. If its height has changed, then, by property 3, it must have *reduced* height by at least 1.

Now consider any maximum pile that does not begin at the root of $\mathcal{T}_H$. Sometime in the first $3 \cdot N$ steps, it will reach the root of $\mathcal{T}_H$. At that time, either it has changed size (and so by property 3 has had its size reduced by at least 1) or it has not. If it has not changed size, then by the argument above its size must be reduced by at least 1 in the *next* $3 \cdot N$ steps.

Thus after $2 \cdot 3 \cdot N$ steps, every maximum must have had its size reduced by at least 1. Since by property 3 the minimum cannot be reduced, the global discrepancy must have been reduced by at least 1. □

5 Conclusion

In this paper we have introduced and sucessfully analysed a dimension-exchange algorithm for token distribution on complete binary trees. The benefits of the dimension-exchange approach, in that it is extremely simple, uses only locally-available information and is completely scalable, cannot be overstated. We have presented an analysis of the algorithm which indicates that it does indeed provide a solution to $TD(\mathcal{T}_H; \Delta, M, \delta)$ for $\Delta > H$, $\Delta > \delta \geq H$. Furthermore, we have shown that under the action of the algorithm, the discrepancy of the underlying token distribution converges to this limit in time linear to the original discrepancy. Specifically, problem $TD(\mathcal{T}_H; \Delta, M, \delta)$ is solved in time $\mathcal{O}(N \cdot (\Delta - \delta))$. In doing so, this paper is the first to show that the dimension-exchange technique, an important class of data distribution protocols, can lead to optimal token distribution algorithms on tree-connected architectures.

References

[Cyb89] G. Cybenko. Dynamic load balancing for distributed memory multiprocessors. *Journal of Parallel and Distributed Computing*, 7:279–301, 1989.

[DG94] D. Diderich and H. Gengler. An efficient algorithm for solving the token distribution problem on k-ary d-cube networks. Technical Report LITH-97, Swiss Federal Institute of Technology, Lausanne, Switzerland, March 1994.

[HLM+90] S.H. Hosseini, B. Litow, M. Malkawi, J. McPherson, and K. Vairavan. Analysis of graph coloring based distributed load balancing algorithm. *Journal of Parallel and Distributed Computing*, 10:160–166, 1990.

[HT96] M.E Houle and G. Turner. Dimension exchange token distribution on the mesh and torus. *Lecture Notes in Computer Science: Proceedings of ISAAC'96*, 1996.

[Lei92] F. Thompson Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes.* Morgan Kaufmann Publishers, San Mateo, California, 1992.

[MadHOW93] F. Meyer auf der Heide, B. Oesterdiekhoff, and R. Wanka. Strongly adaptive token distribution. In *Proceedings of the 20th ICALP*, pages 398–409, 1993.

[Pla89] C. G. Plaxton. Load balancing, selection and sorting on the hypercube. In *Proceedings of the ACM-SPAA*, pages 64–73, 1989.

[PU87] D. Peleg and E. Upfal. The generalised packet routing problem. *Theoretical Computer Science*, 53:218–293, 1987.

[RWS88] S. Ranka, Y. Won, and S. Sahni. Programming a hypercube multicomputer. *IEEE Software*, pages 69–77, 1988.

[TS95] G. Turner and H. Schröder. Token distribution on reconfigurable d-dimensional meshes. In *Proceedings of the 1st IEEE International Conference on Algorithms and Architectures for Parallel Processing*, volume 1, pages 335–344, 1995.

[TT97] Ewan Tempero and Gavin Turner. Dimension-exchange token distribution on complete binary trees. *Communications of the Australasian Computer Society*, 19(1), 1997. Presented at Computing: The Australasian Theory Symposium 1997. Also appears as VUW Technical Report CS-TR-96/22.

[XL92a] C.Z. Xu and F.C.M. Lau. Analysis of the generalized dimension exchange method for dynamic load balancing. *Journal of Parallel and Distributed Computing*, 16:385–393, 1992.

[XL92b] C.Z. Xu and F.C.M. Lau. The generalized dimension exchange method on some specific structures. Technical Report TR-92-02, University of Hong Kong, 1992.