Aspects of k-k Routing in Meshes and OTIS Networks.

Dissertation zur Erlangung des akademischen Grades

Doctor rerum naturalium

(Dr. rer. nat.)

vorgelegt der Fakultät Informatik und Automatisierung

der Technischen Universität Ilmenau

von

Dipl. Inform. Andre Osterloh

Gutacher:

1. Univ.-Prof. Dr. Manfred Kunde, Technische Universität Ilmenau
2. Univ.-Prof. Dr. Michael Kaufmann, Universität Tübingen
3. Priv.-Doz. Dr. Peter Rossmanith, Technische Universität München

vorgelegt am:

10. Juni 2002

verteidigt am:

19. September 2002

Abstract

Efficient data transport in parallel computers build on sparse interconnection networks is crucial for their performance. A basic transport problem in such a computer is the k-k routing problem. In this thesis, aspects of the k-k routing problem on r-dimensional meshes and OTIS-G networks are discussed. The first oblivious routing algorithms for these networks are presented that solve the k-k routing problem in an asymptotically optimal running time and a constant buffer size. Furthermore, other aspects of the k-k routing problem for OTIS-G networks are analysed. In particular, lower bounds for the problem based on the diameter and bisection width of OTIS-G networks are given, and the k-k sorting problem on the OTIS-Mesh is considered. Based on OTIS-G networks, a new class of networks, called Extended OTIS-G networks, is introduced, which have smaller diameters than OTIS-G networks.

Contents

1 Introduction. 1

1.1 Outline of the Thesis. 3

2 Basic Definitions. 7

2.1 Basic Definitions in Graph Theory. 7

2.2 Definition of the Model. 9

2.2.1 Model of Computation. 9

2.2.2 Routing Problems, Packets and Algorithms. 10

2.3 Embeddings and Emulations. 14

3 Meshes and Basic Problems. 16

3.1 Definitions. 16

3.2 Routing in One-Dimensional Meshes. 18

3.2.1 Embedding into Networks. 19

3.2.2 Routing with Bounds on Arrival Times. 19

3.2.3 Dynamic Routing. 27

3.3 Conclusion. 45

4 Oblivious Routing. 46

4.1 Introduction. 46
4.2 Discussion of Previous Results. 50
4.2.1 The Algorithm Presented in [11, 14]. 50
4.2.2 The Algorithm Presented in [32]. 53
4.3 High Level Structure of the Algorithm. 55
4.4 Oblivious k-k Routing. 56
4.4.1 Oblivious k-k Routing on Networks. 56
4.4.2 Oblivious k-k Routing on r-Dimensional Meshes. . . . 67
4.5 Conclusion. 78

5 OTIS Networks. 80

5.1 Introduction. 80
5.2 Definition of OTIS-G Networks. 82
5.3 A Lower Bound for Routing on OTIS-G Networks. 83
5.4 Sorting on the OTIS-Mesh. 85
5.4.1 Preliminaries 86
5.4.2 Sorting with All-to-All Mappings. 87
5.4.3 Emulation of $M_{4,n}$ by OTIS-$M_{2,n}$. 90
5.4.4 Substructures in the OTIS-Mesh. 93
5.4.5 All-to-All Mapping. 95
5.4.6 The Sorting Algorithm. 101
5.4.7 A Lower Bound for Sorting on the OTIS-Mesh. 103
5.4.8 A Comparison with Mesh Algorithms. 104
5.5 Oblivious Routing on OTIS-G Networks. 105
5.6 Diameter of OTIS-G Networks. 107

5.7 Extended OTIS-G Networks. 115
5.7.1 Basic Definitions and Properties. 115
5.7.2 Shortest Paths in Extended OTIS-G Networks. 116
5.7.3 Diameter of Extended OTIS-G Networks. 122
5.7.3.1 Hypercubes. 123
5.7.3.2 Rings. 126
5.7.3.3 One-dimensional Meshes. 136
5.7.3.4 Two-dimensional Meshes. 140
5.8 Conclusion. 142
Bibliography 143
Index 150
Appendix 157
Theses of the dissertation. 157
Zusammenfassung. 159
Erklärung. 160

List of Figures

2.1 A packet used in routing. 11

3.1 The structure of $M_{1,5}$. 17
3.2 The structure of $M_{2,4}$. 17
3.3 An example for the problem considered in Section 3.2.3. . . . 27
3.4 A one-dimensional mesh of size m with m injectors. 27

4.1 A $(4,4)$-partition of size $(4,4,3)$ of $M_{2,4}$. 69
4.2 Path of a packet in oblivious routing. 70
4.3 Source and destination blocks in $M_{3,15}$. 74
4.4 Source blocks and exit nodes. 75
4.5 Connection paths from $exit_{(i,2)}$ to entry nodes. 76
4.6 Trees in a partitioning of $M_{3,16}$. 76

5.1 The structure of OTIS-$M_{2,2}$. 84
5.2 An OTIS-G network divided in two areas. 85
5.3 Superblocks, blocks, and bricks in an OTIS-Mesh network. . . 94
5.4 An OTIS-Mesh divided in two areas X and Y. 103
5.5 Extended OTIS-G networks, $|V_G| = 2$. 123

5.6 HCN(2,2). 124

5.7 The structure of $X_{R_3,f}$. 127

5.8 The structure of X_{R_4,f_4}. 128

5.9 Mirror property of f_n. 129

5.10 Cases for the proof of Lemma 5.38. 132

5.11 Cases for the proof of Lemma 5.39. 134

5.12 The structure of $X_{M_{1,4},f_4}$. 137

5.13 The structure of $X_{M_{1,5},f_5}$. 137

5.14 The structure of $X_{M_{2,3},f_3}$ (without optical links). 141

Chapter 1

Introduction.

Parallel computers consist of (two or more) processors. To solve problems efficiently these processors have to communicate with each other. There are different communication methods possible, e.g.

- *Communication via shared-memory.* A theoretical parallel computer model, in which communication is based on shared-memory, is referred to as PRAM (parallel random access machine). A PRAM consists of a global memory that is uniformly accessible to all processors. There exists a global clock, enabling the processors to execute instructions in a synchronous way. Communication among the processors is done using the global memory.

- *Communication via links.* A parallel computer is modeled by a synchronized network of processors connected by links. In such a network, a direct communication between two processors is only possible if they are connected by a link. For the communication of two non-connected processors, data has to be transported through the network via a path of directly connected processors.

In this thesis, parallel computers based on communication via links are considered. Obviously, the efficiency of the communication depends on the underlying connection network. Ideally, each processor is connected to any other processor in the network, i.e., a complete graph is used as connection network. Such a network would need $O(n^2)$ links to connect n processors. This is considered infeasible, since such a completely connected network would be far too expensive, even for a small number of processors. A small constant number of links per processor would be feasible. In this work, two kinds of connection networks are considered, r-dimensional meshes and OTIS-G networks. The number of links per processor is bounded by $2r$ in r-dimensional meshes and by $d_G + 1$ in OTIS-G networks, where d_G is the number of links per processor in network G.

Especially for meshes the problem of efficient communication between processors has been studied intensively in the last years. One of the best studied communication problem is the problem where each processor has to send and receive at most k packets, the *k-k routing problem*. In the last ten years, many variants of this problem were solved efficiently [7]. For example, the problem where each processor sends and receives exactly one packet, the so called *permutation routing problem*, was solved on a two-dimensional mesh with n processors in each dimension, in $2n - 2$ steps and buffer size 32 [46]. This is the optimal number of steps for the permutation routing problem on a two-dimensional mesh, i.e., the number of steps can not be reduced any further. Nevertheless, some aspects of the k-k routing problem remain unsolved. One is the problem of designing an *oblivious* routing algorithm with a small buffer size for the r-dimensional mesh that solve the k-k routing problem in a number of steps close to the best known lower bound. Even the order of magnitude of the number of steps used by the best known oblivious algorithms does not come close to the lower bound for the case of meshes of

a dimension greater than two.

In an oblivious routing algorithm the path of a packet through the network only depends on its source and destination processor within the network and hence is indepent of the path of other packets. This property of oblivious algorithms is interesting, since it allows one to design simple and hence practical algorithms.

This work offers a substantial contribution to solving the problem. It presents oblivious routing algorithms that solve the k-k routing problem on the considered networks in an asymptotically optimal number of steps and with a small buffer size ($O(k)$), i.e., the number of steps achieved by these algorithms differs from the optimal number by at most a constant factor.

OTIS networks have not been given the same attention as meshes. In these networks electronical and optical links are used to connect the processors. Additionally to oblivious algorithms, this thesis investigates some further aspects of the k-k routing problem as the diameter and the bisection width of the networks. Furthermore, a communication problem very similar to the k-k routing problem is considered, the k-k sorting problem.

1.1 Outline of the Thesis.

In Chapter 2, basic definitions are presented. Shortly reviewing basic definitions in graph theory, the problems and model of computation under consideration are introduced. The chapter concludes with a definition of embeddings which are used in Chapter 3, Chapter 4 and Section 5.4 to obtain the results.

In Chapter 3, r-dimensional meshes are defined and two problems on one-dimensional meshes are solved. It is shown how to employ these results to obtain solutions for r-dimensional meshes and other networks. Both

problems discussed in this chapter play an important role in the design of oblivious k-k routing algorithms for r-dimensional meshes in Chapter 4.

Chapter 4 is concerned with oblivious routing. It is shown that k-k routing can be solved obliviously on a large class of networks of fixed degree (the degree of such a network is independent of its size) in an asymptotically optimal number of steps with buffer size $O(k)$. An oblivious algorithm is presented that solves the k-k routing problem on networks, for which a special partitioning exists, and the result is applied to r-dimensional meshes. It is shown that a deterministic and oblivious routing algorithm for r-dimensional meshes of side length n exists that solves the k-k routing problem in $O(kn^{\frac{r}{2}})$ steps with buffer size $O(k)$. For $r > 2$ and all k, the order of magnitude of the running time is smaller than those of other deterministic and oblivious algorithms with buffer size $O(k)$ known before. For the case $r = 2$ and $k = 1$, algorithms with an asymptotically optimal running time are known [14, 32] and discussed in this chapter.

In Chapter 5, aspects of k-k routing on OTIS networks are discussed, organized in two parts. The first part, Sections 5.2-5.6, deals with OTIS-G networks, the second part introduces Extended OTIS-G networks.

In Section 5.3 a lower bound for routing on OTIS-G networks is proved. An OTIS-G network is a kind of hierarchical network. Its structure depends on the structure of the graph G. An OTIS-G network, where G is a two-dimensional mesh, is called an OTIS-Mesh. In Section 5.4, algorithms solving (full) k-k sorting problems on the OTIS-Mesh are presented. It is shown how the technique of solving k-k sorting problems by all-to-all mappings can be used to solve the problem on OTIS-Meshes. Rounding off Section 5.4, lower bounds for k-k routing and k-k sorting on the OTIS-Mesh are given and the obtained sorting algorithm is compared with sorting algorithms designed for meshes. In Section 5.5, the results of Chapter 4 are applied to

OTIS-G networks in order to obtain oblivious routing algorithms that solve the k-k routing problem with $O(k)$ buffer size. For all graphs G of fixed degree, an asymptotically optimal running time is achieved. In Section 5.6, the diameter of OTIS-G networks is determined. This leads to the definition of Extended OTIS-G networks, where a few links are added to reduce the diameter. Definition and some basic properties of Extended OTIS-G networks are given in Section 5.7.1. Section 5.7.2 is concerned with determining shortest paths in Extended OTIS-G networks. Finally, in Section 5.7.3 the diameter for several Extended OTIS-G networks is determined.

The following tables give a short summary of the most important results and can be used as a guide through this thesis.

Oblivious routing on networks.					
type	network	buffer size	steps	lower bound	reference
full 1-1	$M_{2,n}$	10	50n	2n-2	Section 4.4.2
k-k	$M_{r,n}$[1], $r > 1$	$k + 9$	$O(kn^{r/2})$	$\Omega(kn^{r/2})$	Section 4.4.2
k-k	$\mathcal{N} = (V, E)$[2]	$O(k)$	$O(k\sqrt{\lvert V\rvert})$	$\Omega(k\sqrt{\lvert V\rvert})$	Section 4.4.1
k-k	OTIS-G network[2] $G = (V, E)$	$O(k)$	$O(k\lvert V\rvert)$	$\Omega(k\lvert V\rvert)$	Section 5.5

[1] r-dimensional mesh of side length n

[2] some additional conditions have to be fulfilled

Aspects of k-k routing in OTIS-G networks, $G = (V, E)$.			
type	network	result	reference
lower bound	general G	$\max\{2D(G) + 1, \frac{k}{bw(G)}\}$	Section 5.3
k-k sorting	$G = M_{2,n}$	buffer: $k + 4$, steps: $\max\{8n+o(n), 2kn+o(kn)\}$	Section 5.4.6
obl. k-k	G fixed degree	buffer: $O(k)$, steps: $O(k\|V\|)$	Section 5.5
diameter	general G	$2D(G) + 1$	Section 5.6
diameter	Extended OTIS	reduction, $< 2D(G) + 1$	Section 5.7.3

$D(G)$ diameter of graph G, $bw(G)$ bisection width of graph G

Chapter 2

Basic Definitions.

In this chapter we provide basic definitions and notations used throughout this thesis. Special notations will be given in the chapters where they are needed.

The set of integers is denoted by $\mathbb{Z}$. $\mathbb{N}$ is the set of natural numbers, without zero, $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$, and $[n] = \{0, \ldots, n-1\}$. The cardinality of a set M is denoted by $|M|$. The set of all subsets of M of cardinality two is written as $\mathcal{P}_2(M)$. For a function $f : A \longrightarrow B$ and $U \subseteq B$ the inverse image of U is denoted by $f^{-1}(U)$. The set $f^{-1}(U)$ consists of all elements $a \in A$ such that $f(a) \in U$.

2.1 Basic Definitions in Graph Theory.

Graphs are a very important concept that is used throughout this thesis. For definitions that can not be found here we refer the reader to any book that gives an introduction into graph theory, e.g. [56, 36].

An undirected graph $G = (V, E)$ consists of a finite set of *nodes* V and a finite set of *edges* E, where each edge $e \in E$ is an element of $\mathcal{P}_2(V)$. The

size of a graph is the number of its nodes. Two nodes connected by an edge are called *adjacent*. An edge $e = \{x, y\} \in E$ is *incident* to x and y. All nodes adjacent to a node $x \in V$ are *neighbours* of x. The *degree* $deg_G(x)$ *of a node* $x \in V$ is the number of its neighbours. The *degree* $deg(G)$ *of a graph* G is the maximal number of neighbours of a node in G. If all nodes in G have the same degree, then G is called *regular* or $deg(G)$*-regular*. A family of graphs has *fixed degree* if a constant c exists such that all graphs of the family have at most degree c.

A *path* p between nodes x and y in G is a sequence

$$p = \{u_0, u_1\}, \{u_1, u_2\}, \ldots, \{u_{l-1}, u_l\}$$

of edges such that $u_0 = x$ and $u_l = y$. Path p *uses* edge $e \in E$ if $e = \{u_i, u_{i+1}\}$ for $i \in [l]$. Path p *uses* node $x \in V$ if $x = u_i$ for $i \in [l]$. The set of nodes used by a path is denoted by $V(p)$. The *length* $|p|$ of a path p is the number of its edges. We further stipulate that for each node an empty path of length zero between x and x exists. A path in which each node is visited at most once, i.e. $u_i \neq u_j$ for $0 \leq i < j \leq l$, is called *simple*. A simple path p in G such that $V(p) = V$ is called a *Hamiltonian path*. If $u_0 = u_l$ and $|p| > 0$, then p is called a *cycle*. An undirected graph G is *connected* if for all nodes $x, y \in V$ a path in G between x and y exists. For all nodes $x, y \in V$ in a connected graph, let $d_G(x, y)$ denote the *distance* of x and y, i.e. the length of a shortest path between x and y in G. The *diameter* $D(G)$ of G is $\max\{d_G(x, y) \mid x, y \in V\}$.

For $X, X' \subseteq V$ let $C_G(X, X')$ be the number of edges e in E such that $e \cap X \neq \emptyset$ and $e \cap X' \neq \emptyset$. The *bisection width* $bw(G)$ of G is $\min\{\frac{C_G(X, V-X)}{|X|} \mid X \subseteq V, |X| = \lfloor |V|/2 \rfloor\}$.

In a *directed graph* $G = (V, E)$ every edge is directed from one node to another, i.e. $E \subseteq V \times V$. We denote a *directed edge* e from node x to node

y by $e = (x, y)$. We say that e is incident to x and y (and x, y are incident to e). Paths and cycles in directed graphs are defined analogously to the undirected case (for directed graphs we speak of paths from x to y). The *distance* $d_G(x, y)$ from x to y is the length of a shortest path from x to y, if a path from x to y exists and ∞ else. The diameter of a directed graph is defined as in the undirected case.

A directed graph G is *connected* if for all nodes $x, y \in V$, a path in G from node x to node y or a path from node y to node x exists. A *tree* $G = (V, E)$ with *root* $r \in V$ is a directed graph (V, E) without cycles such that for all $x \in V$ a path from r to x exists. We denote a tree with root r by a triple (V, E, r).

The *directed version* $\vec{G}$ of an undirected graph $G = (V, E)$ is a directed graph with node set V and edge set $\vec{E}$. The set $\vec{E}$ can be obtained by replacing every undirected edge $e = \{x, y\} \in E$ by two directed edges (x, y) and (y, x).

Unless explicitly mentioned, we assume in this work that G is undirected, connected and that it contains at least one node. If the node or edge set of a graph G is not given explicitly, we will use V_G to denote the node set and E_G to denote the edge set of G.

2.2 Definition of the Model.

2.2.1 Model of Computation.

A *network* can be described by an undirected and connected graph $\mathcal{N} = (V, E)$. The set of nodes V represents the set of processors and the set E of of edges represents the set of *communication links*. Each edge $e = \{x, y\} \in E$ represents a communication link between processors x and y.

The processors operate in a *synchronous* fashion, and communicate by send-

ing *packets* over the communication links. To model the tranport of packets we use the so called *store-and-forward packet routing model*. In this model data is organized in packets and it is not allowed to send data not attached to a packet. The packets are atomic entities, i.e., a packet must be stored completely in a processor before it can be send to the next processor. Other routing models, e.g., the *wormhole routing model* or *cut-through routing model*, allow that a packet is partitioned into *flits* and spread out across one ore more nodes. For an overview of routing models see [31].

In a single *step*, a processor receives a number of packets that were sent to it by neighbouring processors in the previous step, perform some amount of internal computation, and send a number of packets across its communication links to neighbouring processors. Packets that are received and not send in the same step have to be stored in a *buffer* on the processor. For the internal computations a processor possesses a processing unit and a local memory and has access to the packets stored in the buffer or received from neighbouring processors.

The *bandwidth* of a communication link is defined as the number of packets that can be transmitted over the link in either direction in a single step. In some parts of this work we allow that the links of the network have different bandwidths. In this case, we will mention it explicitly. Unless explicitly mentioned, we assume that the bandwidth of a link is one.

2.2.2 Routing Problems, Packets and Algorithms.

In this work, we consider aspects of *routing*. We analyse *packet routing problems*. A packet routing problem on a network $\mathcal{N} = (V, E)$ is the problem of rearranging a set of packets in $\mathcal{N}$ such that every packet ends up at the processor specified by its destination address. A packet routing problem on network $\mathcal{N}$ can be described by a triple $(\mathcal{P}, src, dst)$, where $\mathcal{P}$ is a set

Figure 2.1: A packet used in routing.

of packets, and $src, dst : \mathcal{P} \longrightarrow [|V|]$ are mappings. For a packet $p \in \mathcal{P}$, $src(p)$ and $dst(p)$ are *addresses* of processors. The address of a processor in $\mathcal{N}$ is determined by a fixed bijection $\mathcal{I} : V \longrightarrow [|V|]$. In a packet routing problem $(\mathcal{P}, src, dst)$ each packet $p \in \mathcal{P}$ is loaded in the processor specified by address $src(p)$ initially and has to be sent to processor specified by address $dst(p)$. We call the processor specified by $src(p)$ *source processor*, *source node*, or *source* of packet p, and the processor specified by $dst(p)$ *destination processor*, *destination node*, or *destination* of packet p.

A k-k *routing problem* is a packet routing problem in which each processor is source and destination of at most k packets. If each processor is source and destination of exactly k packets the problem is called a *full* k-k routing problem. We call a full 1-1 routing problem a *permutation routing problem*.

We assume that each packet consists of four fields (see Figure 2.1), the message field, the source address field, the destination address field, and additional information field. The source and destination address field require $O(\log |V|)$ bits and we restrict the size of the additional information field to $O(\log k|V|)$ bits when we solve k-k routing problems. During routing, we allow that the additional information field of a packet is changed by a processor. All other fields of a packet are not allowed to be changed by a processor.

In this thesis, we consider *deterministic* algorithms. We do not allow that any random decisions are made by an algorithm.

We are interested in the *number of steps* required to route all packets to their

destination in the *worst case*. We call the number of steps an algorithm $\mathcal{A}$ requires to solve a problem in the worst case the *running time* of $\mathcal{A}$. We give *upper* and *lower bounds* for this value. Lower bounds for routing problems are given by the diameter and the bisection width of a network, e.g., algorithms that solve k-k routing problems on a network $\mathcal{N}$ have at least a running time of $\max\{\frac{k}{bw(\mathcal{N})}, D(\mathcal{N})\}$. We call these bounds *diameter* and *bisection* bound.

Besides the running time, the *buffer size* an algorithm needs to solve a packet routing problem is an important measure of its performance. We define the buffer size of an algorithm as the maximal number of packets that are located in any processor during the execution of the algorithm. This includes packets that want to pass the processor. We assume that a packet is absorbed when it reaches its destination processor. In the literature, there are several different definitions for the buffer size of an algorithm. In one definition (see e.g. [44]) each link of a processor has a *link buffer*, where packets can be stored, and an additional buffer called *injection buffer*. The task of the injection buffer is to store all packets for which the processor is source. In this definition, the buffer size of an algorithm is defined as the maximal number of packets in a link buffer. An algorithm for k-k routing that has buffer size c under this definition can have a buffer size of up to $k + deg(\mathcal{N})c$ under our definition. In another definition (see e.g [46]), the size of link buffers is restricted to one, but the processors have an *internal buffer* to store packets. The buffer size of an algorithm under this definition is defined as the maximal number of packets in any internal buffer. An algorithm that has buffer size c under this definition can have a buffer size of up to $c+deg(\mathcal{N})$ under our definition. We consider k-k routing algorithms that have a bounded buffer size, i.e., the buffer size is $O(k)$.

Another important aspect of an algorithm is its *simplicity*. One can hope

that a simpler algorithm will be more practical. Unluckily it is very hard to find a measure for the simplicity of an algorithm. Measures of simplicity for an algorithm could be:

- The buffer size of an algorithm. Has it a buffer size of $O(1)$, or not?
- The kind of paths used by the algorithm. E.g. are shortest or simple paths used?
- The kind of routing strategy used by the algorithm. E.g. is the routing algorithm adaptive or oblivious? Depends the path of a packet on other packets or not?
- The control structure of an algorithm. Are only simple calculations needed for the routing decision?

There are several other measures for simplicity possible. We consider in this thesis *oblivious* routing algorithms. In the literature a formal definition of an oblivious routing algorithm is hard to find. Very often an oblivious routing algorithm is described as an algorithm where the path of a packet only depends on its source and destination and is independent of other packets in the network [49, 22, 41, 17, 32, 12]. Such a definition let room for interpretations. We give a formal definition of an oblivious algorithm.

An element of a set X of packet routing problems on a network $\mathcal{N}$ is called an *instance* of X on $\mathcal{N}$ and an algorithm that solves all instances of X on $\mathcal{N}$ is an algorithm for X on $\mathcal{N}$. In the case that X is the set of all k-k routing problems on $\mathcal{N}$, an algorithm for X on $\mathcal{N}$ is called an k-k routing algorithm on $\mathcal{N}$. If it is clear which network is meant, $\mathcal{N}$ is omitted.

Definition 2.1 *Let $\mathcal{N} = (V, E)$ be a network and $Path(\mathcal{N})$ be the set of all paths in network $\mathcal{N}$. Let X be a set of packet routing problems on $\mathcal{N}$ and $\mathcal{A}$ be a deterministic algorithm for X on $\mathcal{N}$. Let*

$$path(\mathcal{A}, (\mathcal{P}, src, dst)) : \mathcal{P} \longrightarrow Path(\mathcal{N})$$

be a mapping such that $path(\mathcal{A}, (\mathcal{P}, src, dst))(p)$ *is the path on which a packet* $p \in \mathcal{P}$ *is sent from* $src(p)$ *to* $dst(p)$ *by* $\mathcal{A}$ *when algorithm* $\mathcal{A}$ *solves the instance* $(\mathcal{P}, src, dst)$ *of* X *on* $\mathcal{N}$.

Algorithm $\mathcal{A}$ *is called an oblivious routing algorithm for* X *on* $\mathcal{N}$*, if and only if a mapping*

$$\pi : V \times V \longrightarrow Path(\mathcal{N})$$

exists, such that for all instances $(\mathcal{P}, src, dst)$ *of* X *on* $\mathcal{N}$ *the following holds:*

$$\forall p \in \mathcal{P} : \pi(src(p), dst(p)) = path(\mathcal{A}, (\mathcal{P}, src, dst))(p).$$

2.3 Embeddings and Emulations.

There are two different kinds of strategies to emulate a network $\mathcal{N}_1$ by a network $\mathcal{N}_2$. One possible strategy is to use *static* embeddings. In this case every node in $\mathcal{N}_1$ is simulated by a fixed set of nodes of $\mathcal{N}_2$. The other possible strategy is to use *dynamic* embeddings. In this case, in every step, every node in $\mathcal{N}_1$ is simulated by at least one node of $\mathcal{N}_2$. Although dynamic embeddings are known to be more powerful than static embeddings (see [1, 21]) we will use static embeddings since they are sufficient for our purpose. Furthermore, in a static embedding, the path a packet travels in $\mathcal{N}_2$ to simulate a step of $\mathcal{N}_1$ depends not on the paths of other packets. We need this property in this work in Chapter 4. For an overview of results for static embeddings see [38].

A static embedding of a network $\mathcal{N}_1$ into a network $\mathcal{N}_2$ is a mapping

$$\Phi : \mathcal{N}_1 \longrightarrow \mathcal{N}_2$$

that maps nodes of $\mathcal{N}_1$ to nodes of $\mathcal{N}_2$ and edges of $\mathcal{N}_1$ to paths in $\mathcal{N}_2$. The *dilation* of an embedding is defined as the longest path $\Phi(e)$ where e is an edge of $\mathcal{N}_1$, i.e., the dilation of Φ is

$$\max\{|\Phi(e)| : e \in E_{\mathcal{N}_1}\}.$$

The *congestion* of an embedding is the maximal number of paths $\Phi(e)$ that uses an edge of $\mathcal{N}_2$, i.e., the congestion of Φ is

$$\max\{|\{\Phi(e) : e \in E_{\mathcal{N}_1},\ \Phi(e) \text{ uses } e'\}| : e' \in E_{\mathcal{N}_2}\}.$$

Finally, the *load* of an embedding is the maximal number of nodes of $\mathcal{N}_1$ mapped to a node in $\mathcal{N}_2$, i.e., the load of Φ is

$$\max\{|\Phi^{-1}(\{v\})| : v \in V_{\mathcal{N}_2}\}.$$

It is well known that, if an embedding of $\mathcal{N}_1$ in a network $\mathcal{N}_2$ with congestion c, dilation d, and load 1 exists, an emulation of $\mathcal{N}_1$ by $\mathcal{N}_2$ with *slowdown* $O(c+d)$ exists, i.e., any T steps in $\mathcal{N}_1$ can be simulated in $O((c+d)T)$ steps by $\mathcal{N}_2$ [30, 44]. Furthermore, if $\mathcal{N}_1$ and $\mathcal{N}_2$ are of fixed degree, then any communication step in $\mathcal{N}_1$ can be simulated in $O(c+d)$ steps by $\mathcal{N}_2$ using only constant buffer size [44]. Hence we get

Theorem 2.2 ([30, 44]) *Given two networks $\mathcal{N}_1$ and $\mathcal{N}_2$ of fixed degree. If an embedding of a network $\mathcal{N}_1$ in a network $\mathcal{N}_2$ with congestion $O(1)$, dilation $O(1)$, and load 1 exists, then any algorithm which needs T steps and buffer size B on $\mathcal{N}_1$ can be performed by $\mathcal{N}_2$ in $O(T)$ steps with buffer size $B + O(1)$.*

Chapter 3

Meshes and Basic Problems.

The family of mesh-connected networks is one of the most investigated family of networks. Among other preferences the simple structure of mesh-connected networks match the physical constrains for processor layout which makes them interesting for the practice (e.g. J-Machine, Cray T3D). Furthermore, the simple structure allows an efficient implementation of parallel algorithms.

3.1 Definitions.

Definition 3.1 (r-dimensional mesh) *Let $n, r \in \mathbb{N}$. The mesh $M_{r,n}$ is a graph with node set $[n]^r$ and edge set*

$$\{\{(x_0, \ldots, x_{r-1}), (y_0, \ldots, y_{r-1})\} \mid \forall i \in [r] : x_i, y_i \in [n], \sum_{i=0}^{r-1} |x_i - y_i| = 1\}.$$

An edge where $|x_i - y_i| = 1$ for $i \in [r]$ is called an edge of the i-th dimension. The mesh $M_{r,n}$ is called an r-dimensional mesh of side length n. Additionally, $M_{1,n}$ is called a one-dimensional mesh of size n.

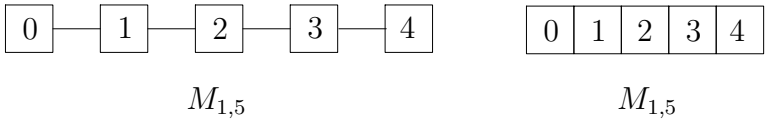

Figure 3.1: The structure of $M_{1,5}$.

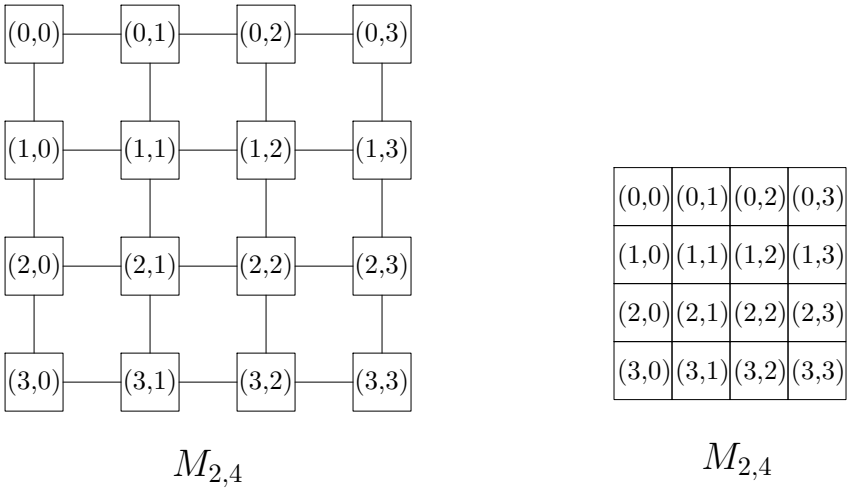

Figure 3.2: The structure of $M_{2,4}$.

For all $r \in \mathbb{N}$, we call $\{M_{r,n} \mid n \in \mathbb{N}\}$ the family of r-dimensional meshes. The family of r-dimensional meshes is of fixed degree.

Figure 3.1 presents two one-dimensional meshes of side length five. In the left one the nodes are represented as boxes and the edges are represented as lines. In the right one the edges of the mesh are not shown. In the following we often use this style to present meshes. Figure 3.2 shows mesh $M_{2,4}$ in both styles.

For the analysis of the algorithms presented in this section we need the following definition of intervals.

Definition 3.2 (interval) *Let $a, b \in \mathbb{N}_0, a \leq b$. The interval from a to b*

consists of steps $a, a+1, \ldots, b-1, b$ *and is denoted with* $[a, b]$. *If we want to exclude the first step, the last step, or the first and the last step of* $[a, b]$, *we write* $(a, b]$, $[a, b)$, *or* (a, b), *respectively.*

3.2 Routing in One-Dimensional Meshes.

In routing algorithms for higher-dimensional meshes, routing on one-dimensional submeshes is often used as a subroutine, since problems on the one-dimensional mesh can be solved efficiently.

For example, given a one-dimensional mesh of size n, it is well known that any distribution of packets over the n nodes can be routed to their destination in the optimal number of steps by a deterministic oblivious routing algorithm using the farthest destination first queueing strategy (e.g. see [47]).

In this section, we analyse algorithms on one-dimensional meshes which are used as subroutines for algorithms in higher-dimensional meshes or other networks later in this thesis. The problems solved by these algorithms differ a little bit from standard routing problems.

In a standard *static* routing problem all packets are stored in buffers in the processors initially. The task is to transport them to their destination as fast as possible using a bounded amount of buffer size.

We consider a problem where packets are created during the routing process. Such problems are called *dynamic* routing problems, e.g. see [4, 15]. We further consider a static problem where a lower bound for the time difference between the arrival of two packets at their destination is given.

The algorithms designed in this section route the packets on the shortest paths to their destination. For any two nodes in an one-dimensional mesh exactly one shortest path between these nodes exists. Hence the algorithms

are oblivious algorithms.

To be able to use the algorithms on other networks we consider embeddings of one-dimensional meshes into networks.

3.2.1 Embedding into Networks.

It is well known that a one-dimensional mesh of size n can be embedded into any network of the same size with dilation $O(1)$, congestion $O(1)$ and load 1 [45, 18, 31]. In [31], Leighton gives an embedding of a one-dimensional mesh of size n in a network of size n with dilation 3, congestion 2, and load 1. Theorem 2.2 yields

Theorem 3.3 *Any algorithm that needs T steps and has buffer size B on an one-dimensional mesh of size n can be performed in $O(T)$ steps and buffer size $B + O(1)$ on any network of size n and fixed degree.*

3.2.2 Routing with Bounds on Arrival Times.

In this section we solve routing problems $(\mathcal{P}, src, dst)$ on a one-dimensional mesh of size m, where $|src^{-1}(\{i\})| \leq k$, for all nodes $i \in [m]$. Every packet $p \in \mathcal{P}$ belongs to a class $c(p) \in \mathbb{N}_0$.

We define

$$\mathcal{P}_x \stackrel{\text{def}}{=} \{p \in \mathcal{P} \mid c(p) = x\},\ x \in \mathbb{N}_0,$$

$$|x| \stackrel{\text{def}}{=} |\mathcal{P}_x|,\ x \in \mathbb{N}_0, \text{ and}$$

$$\mathcal{C}_x \stackrel{\text{def}}{=} \{c(p) \mid p \in src^{-1}(\{x\})\},\ x \in [m].$$

Let d be a mapping $d : \mathbb{N}_0 \longrightarrow \mathbb{N}$ that describes the minimal time difference between the arrival of two packets of the same class at their destination. Here we restrict our attention to problems where each packet leaves the

network through node $m-1$. If two packets of class c leave the network through node $m-1$ in steps t_1 and t_2, then $|t_1 - t_2| \geq d(c)$. To model this, we introduce an additional virtual node m and let m be the destination of all packets. We assume that a directed edge from node $m-1$ to node m exists and that packets are absorbed if they reach node m. This results in the following problem.

Problem 1: Gap Routing. Given is a mapping $d : \mathbb{N}_0 \longrightarrow \mathbb{N}$, $k \in \mathbb{N}_0$, and a packet routing problem $(\mathcal{P}, src, dst)$ on a one-dimensional mesh of size m equipped with an additional virtual node m. For all $p \in \mathcal{P}$ the value $c(p)$ is stored in the additional information field of p. All nodes $i \in [m]$ know $d(c(p))$ for all $p \in src^{-1}(\{i\})$.

A routing problem is a *gap routing problem* if and only if the following holds:

P1.1 $\forall p \in \mathcal{P} : dst(p) = m$,

P1.2 $\forall i \in [m] : k_i \stackrel{\text{def}}{=} |src^{-1}(\{i\})| \leq k$, and

P1.3 $\forall c \in \mathbb{N}_0, \forall p_1, p_2 \in \mathcal{P}_c$, $p_1 \neq p_2$: If p_1 and p_2 reach m in steps t_1 and t_2, then $|t_1 - t_2| \geq d(c)$.

□

First we show a lower bound.

Lemma 3.4 *If $|\mathcal{P}| > 1$, then any algorithm solving an instance of* **Problem 1** *needs at least* $\max\{(a), (b)\}$ *steps, where*

(a)$= m - \max\{i \mid i \in [m],\, k_i > 0\} + \sum_{i \in [m]} k_i$, *and*

(b)$= \max\{m - \max\{i \mid \mathcal{P}_c \cap src^{-1}(\{i\}) \neq \emptyset,\, i \in [m]\} + (|c| - 1)d(c) : |c| > 0,\, c \in \mathbb{N}_0\}$.

Proof:

All packets have destination m. Hence all packets have to use the edge from node $m-1$ to node m. This needs at least $\sum_{i\in[m]} k_i$ steps. To use the edge the packets have to reach node m. Thus any algorithm needs at least $m - \max\{i \mid i \in [m],\, k_i > 0\} + \sum_{i\in[m]} k_i$ steps.

Two packets of class c have to arrive at node m with a time difference of at least $d(c)$ steps. Hence the last packet has to arrive at node m at least $(|c|-1)d(c)$ steps after the first one. The first packet of class c can reach node m not before step $\max\{m - \max\{i \mid \mathcal{P}_c \cap src^{-1}(\{i\}) \neq \emptyset,\, i \in [m]\}$. Hence at least $\max\{m-\max\{i \mid \mathcal{P}_c\cap src^{-1}(\{i\}) \neq \emptyset,\, i \in [m]\}+(|c|-1)d(c) : |c| > 0,\, c \in \mathbb{N}_0\}$ steps are needed. ◇

Two packets of the same class have to reach node m with a time difference. To accomplish this, every node i maintains a counter $z_{c,i}$ for each class $c \in \mathcal{C}_i$, where it counts the number of steps since the last packet of class $c \in \mathcal{C}_i$ has left node i. With the help of this counter the node i decides whether a packet is allowed to be sent to node $i+1$ or not.

Definition 3.5 (transportable packet) *A packet p stored on node i is transportable in step t if and only if the following conditions are fulfilled in step t:*

- *No packet enters node i from node $i-1$.*
- $z_{c(p),i} \geq d(c(p))$.

Now we give an algorithm for our problem. We assume that the buffers are numbered. Initially the packets are stored in buffers $1, \ldots, k$.

Algorithm 1:

- Step $t = 0$, initialization:

- $\forall i \in [m]\, \forall c \in \mathcal{C}_i : z_{c,i} := d(c) - i$.

- Step $t \geq 0$:

  1. $\forall i \in [m]$: If packet p enters node i, then send it to node $i+1$. If $c(p) \in \mathcal{C}_i$, then $z_{c(p),i} := 0$.
  2. $\forall i \in [m]$: If no packet enters node i and there is at least one transportable packet on node i, then choose the transportable packet stored in the buffer with the smallest number and send it to node $i+1$. If packet p is sent to node $i+1$, then $z_{c(p),i} := 0$.
  3. $\forall i \in [m]\, \forall c \in \mathcal{C}_i$: If $z_{c,i} < d(c)$, then $z_{c,i} := z_{c,i} + 1$.

To analyse the algorithm the following definition is needed.

Definition 3.6 (rank of a packet) *Let $p, q \in \mathcal{P}_c$. Packet p is initially stored on node i in buffer j and packet q is initially stored on node i' in buffer j'. We define $p \leq_c q :\Longleftrightarrow i < i' \vee (i = i' \wedge j \leq j')$ and $rank(p, \leq_c) \stackrel{\text{def}}{=} |\{q \in \mathcal{P}_c \mid q \leq_c p\}|$. We say a packet p has rank l in class c if and only if $l = rank(p, \leq_c)$.*

Algorithm 1 transports the packets of class c such that the packet of rank i in class c is the i-th packet of class c that is sent to node m. Before this can be proved in Lemma 3.7, we have to introduce some notations.

A node i sends up to $|c|$ packets of class c to node $i+1$. If node i sends $s \leq |c|$ packets $p_1, \ldots, p_s$ of class c to node $i+1$ in steps $t_1 < \ldots < t_s$, then we denote packet p_j, $1 \leq j \leq s$, by (j, c, i).

The following proofs use the value of $z_{c,i}$ in step t. By this we mean the value of $z_{c,i}$ **before** 1. in step t of **Algorithm 1**.

Lemma 3.7 *For all $p \in \mathcal{P}$ the following holds:*

$(src(p){=}i \wedge c(p) = c \wedge j = rank(p, \leq_c)) \Longrightarrow (\forall i',\, i \leq i' < m : p = (j, c, i'))$.

Proof:

Proof by induction on rank j.

$j = 1$:

Let p be a packet such that $src(p) = i$, $c = c(p)$ and $rank(p, \leq_c) = 1$. If $|src^{-1}(\{i'\}) \cap \mathcal{P}_c| > 0$, then $i \leq i' < m$ (definition of $rank$). If $|src^{-1}(\{i\}) \cap \mathcal{P}_c| > 1$, then by the definition of the rank of a packet, no packet of class c is stored on node i in a buffer with a smaller number than p is stored in. Thus $p = (1, c, i)$.

After a packet has begun to travel to its destination, it is not delayed anymore. Hence no packet is able to overtake another packet. Therefore, we can restrict our attention to $i' \in \{i, \ldots, m-1\}$, where $|src^{-1}(\{i'\}) \cap \mathcal{P}_c| > 0$. Assume $|src^{-1}(\{i'\}) \cap \mathcal{P}_c| > 0$, for $i < i' < m$. Due to the initial setting of $z_{c,i}$, the first time when $z_{c,i} \geq d(c)$ is in a step i. Let t_1 be the step when p is sent to node $i+1$. Then for all steps $t \in [i, t_1 - 1]$ a packet is sent to node $i + 1$. These packets enter node i' in steps $[i', t_1 + i' - i - 1]$ and are sent to node $i' + 1$. Due to the initial setting of $z_{c,i'}$, the first time $z_{c,i'} \geq d(c)$ is in step $\geq i'$. Hence no packet of class c is sent to node $i' + 1$ before p is sent to node $i' + 1$. Thus $p = (1, c, i')$.

$j > 1$:

Let $src(p) = i$ and $c(p) = c$ and $rank(p, \leq_c) = j$. Let $p = (x, c, i)$ for $1 \leq x \leq |c|$. By induction hypothesis all packets with rank $< j$ in class c leave i before p. If packets of rank $> j$ in class c are stored in i, then they are sent to node $i+1$ after p (see definition of $rank$ and 2. in the algorithm). Hence $p = (j, c, i)$.

It remains to show that $p = (j, c, i'')$ for $i < i'' < m$. As in the case $j = 1$, we can restrict our attention to $i < i'' < m$, where $|src^{-1}(\{i''\}) \cap \mathcal{P}_c| > 0$.

Let p' be the packet such that $c(p') = c$, $rank(p', \leq_c) = j-1$ and $src(p') = i'$, t_0 be the step when p' is sent to node $i+1$, and t_1 be the step when p is sent to node $i+1$. By induction hypothesis we know $p' = (j-1, c, i)$ and $t_0 < t_1$.

Let p'' be a packet such that $src(p'') = i'' > i$ and $c(p'') = c$ and t_2 be the step when p'' is sent to $i''+1$. We know from the induction hypothesis that $p' = (j-1, c, i'')$, $p = (x, c, i'')$, and $p'' = (x', c, i'')$, where $x, x' > j-1$.

In step t_0 counter $z_{c,i}$ is set to zero (by p'). Hence p can be sent to $i+1$ not before step $t_0 + d(c)$. Hence $t_0 + d(c) \leq t_1$. In all steps $t \in [t_0 + d(c), t_1 - 1]$ a packet $\neq p$ is sent to $i+1$. These packets (we call them blocking packets) reach node i'' in steps $t \in [t_0 + d(c) + i'' - i, t_1 + i'' - i - 1]$ and are sent to node $i''+1$. The packet p' enters node i'' in step $t_0 + i'' - i$ and sets the counter $z_{c,i''}$ to zero. Due to the induction hypothesis we have $t_0 + i'' - i < t_2$. Counter $z_{c,i''}$ is set to zero in step $t_0 + i'' - i$, so $t_0 + i'' - i + d(c) < t_2$. In steps $t \in [t_0 + i'' - i + d(c), t_1 + i'' - i - 1]$ packet p'' is blocked by the blocking packets. In step $t_1 + i'' - i$ packet p enters i'' and is sent to $i''+1$, i.e., p is sent to $i''+1$ after p' is sent and before p'' is sent. Thus $p = (j, c, i'')$. $\diamond$

Lemma 3.8 Algorithm 1 *solves the gap-routing problem. It has a buffer size of* $k+1$.

Proof:

Every packet is routed to node m. Hence the algorithm solves the routing problem. No packet is stored in a node. Hence in any step on any node there are at most $k+1$ packets.

It remains to show that **P1.3** is fulfilled. Let $c \in \mathbb{N}_0$ such that $\mathcal{P}_c \neq \emptyset$. We show the following by induction on $i \in [m]$.

(a) For all $i \in [m]$: If two packets $p_1, p_2 \in \mathcal{P}_c$ are sent to node $i+1$ in steps t_1 and t_2, then $|t_2 - t_1| \geq d(c)$.

P1.3 follows by setting $i = m - 1$. Let p be a packet of rank 1 in class c. Obviously (a) holds for all $i \leq src(p)$.

Now consider a node $i > src(p)$. A node i' exists, such that $src(p) \leq i' < i$ and $\mathcal{P}_c \cap src^{-1}(\{i'\}) \neq \emptyset$. Choose i' maximal.

If $\mathcal{P}_c \cap src^{-1}(\{i\}) = \emptyset$, then (a) follows by an application of the induction hypothesis on i'. If $\mathcal{P}_c \cap src^{-1}(\{i\}) \neq \emptyset$, then choose a packet p' such that $src(p') = i$ and $c(p') = c$ with minimal rank. Let j be the rank of p' in class c. By the induction hypothesis any two packets of rank $< j$ of class c are sent to node $i' + 1$ with a time difference of at least $d(c)$. By Lemma 3.7 these packets are sent to $i + 1$ before packet p' is sent. Due to the counter and the fact that the packets are not stored on their way to node m, node i sends the packets of class c with correct gaps to node $i + 1$. Hence (a) is fulfilled. ◇

Now we analyse the running time of **Algorithm 1**.

Lemma 3.9 (Running Time of Algorithm 1.) *For an arbitrary instance of* **Problem 1** *as defined above* **Algorithm 1** *needs at most*

$$m + \sum_{i \in [m]} k_i + \max\{(d(c) - 1)(|c| - 1) \mid c \in \mathbb{N}_0\}$$

steps.

Proof:

All packets have destination m. Let p be the first packet arriving at node m. By Lemma 3.7 p has rank one in class $c(p)$.

Let i_c be the source node of the packet of rank one in class c. The first time the counter z_{c,i_c} reaches value $d(c)$ is in step i_c. Hence the first packet (call it p') that is sent to a node has source node $\min\{i_c \mid c \in \mathbb{N}_0, |c| > 0\}$. Packet p' is in node $i \in [m]$ in step i. Hence $p = p'$ reaches node m in step m.

Let t_1 be the step when the last packet arrives at node m and let c be its

class. We set $T := [m, t_1]$ and call a step $t \in T$ *empty* if no packet arrives at m in step t. We have $t_1 - m + 1 = \sum_{i\in[m]} k_i + e_t$, where e_t is the number of empty steps in T. T is divided into $|c|$ subintervals $T_0, \ldots, T_{|c|-1}$ by the arrival of packets of class c at m. For $i \in [|c|]$ let $T_i = [t_{0,i}, t_{1,i}]$ such that exactly one packet of class c arrives at m during T_i. We choose T_i such that the packet arrive in the last step, i.e. in step $t_{1,i}$. We further set $t_{0,0} = m$ and $t_{0,i+1} = t_{1,i} + 1$ for $i \in \{1, \ldots, |c| - 1\}$. In T_0 there are no empty steps because $z_{c',i} = d(c')$ for $c' \neq c(p)$ if p (the first packet that arrives at m) is sent to $i+1$. Hence $e_t = 0$, if $|c| = 1$. Now assume that $|c| > 1$. There are e_t empty steps and $|c| - 1$ intervals with empty steps. Hence there exists one $T_i, i > 0$ with at least $\lceil \frac{e_t}{|c|-1} \rceil$ empty steps. A packet of class c is inserted into node i if $z_{c,i} \geq d(c)$. Hence $\lceil \frac{e_t}{|c|-1} \rceil < d(c)$. ◇

Now we consider full problems where $d(c)$ depends on the number of packets of class c. We assume that an upper bound for $d(c)|c|$ exists.

Theorem 3.10 *If $k_0 = k_1 = \cdots = k_{m-1} = k > 0$ and $d(c)|c| \leq m'$ for all $c \in \mathbb{N}_0$, then* **Algorithm 1** *needs at most $m + km + m'$ steps and needs a buffer size of $k + 1$. In this case a lower bound for the running time is $\Omega(km)$. The algorithm uses at most $O(k \log m' + m)$ bits per node for the counter.*

Proof:

We have $\sum_{i\in[m]} k_i = km$. By Lemma 3.4 we obtain the lower bound. For all $c \in \mathbb{N}_0$ we have $(d(c) - 1)(|c| - 1) \leq d(c)|c| \leq m'$. Hence

$$m + \textstyle\sum_{i\in[m]} k_i + \max\{(d(c) - 1)(|c| - 1) \mid c \in \mathbb{N}_0\} \leq m + km + m'.$$

The minimal value of a counter in a node is $-m$. The maximal value is m'. Hence we need at most $O(k \log m' + m)$ bits per node for the counters. ◇

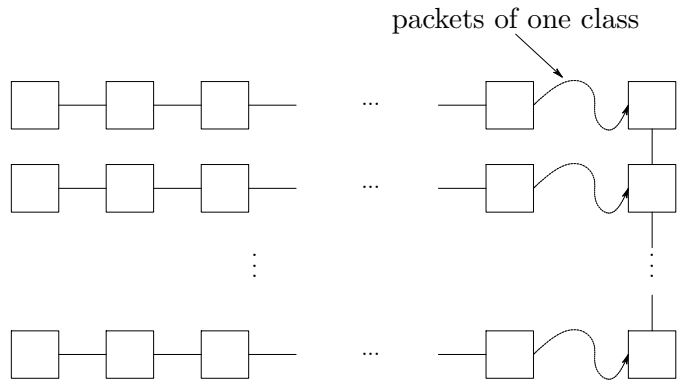

Figure 3.3: An example for the problem considered in Section 3.2.3.

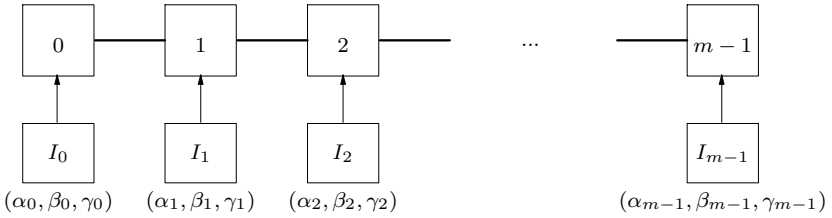

Figure 3.4: A one-dimensional mesh of size m with m injectors.

In the case $m' \in O(km)$ **Algorithm 1** solves instances of **Problem 1** in an asymptotically optimal number of steps.

Corollary 3.11 *If* $k_0 = k_1 = \cdots = k_{m-1} = k > 0$, $m' \in O(km)$, *and* $d(c)|c| \leq m'$ *for all* $c \in \mathbb{N}_0$, $|c| > 0$, *then* **Algorithm 1** *needs* $O(km)$ *steps and has a buffer size of* $k + 1$. *In this case a lower bound is* $\Omega(km)$.

3.2.3 Dynamic Routing.

The problem discussed in this section is motivated by the problem of the previous section. In Section 3.2.2 the algorithm produces a stream of packets in which two packets of the same class reach node m with a certain time

difference. Now assume that such streams of packets are produced in more than one one-dimensional mesh and that all packets of one class have to enter a one-dimensional mesh at different nodes to reach their destination (see Figure 3.3).

We model this by a routing problem on a one-dimensional mesh of size m (see Figure 3.4). For every node $i \in [m]$ there exists an *injector* I_i that creates packets with a certain *creation rate*. An injector I_i is able to insert a created packet into node i. Injectors are known from the analysis of dynamic routing problems [4, 15]. In dynamic routing problems it is often assumed that packets are created with a certain probability and that their destination is chosen according to a specified distribution. Moreover, no bound for the number of created packets is given. Therefore, for dynamic routing problems, the running time of an algorithm is not of interest. For such problems stability, bounds on routing delays and buffers are of interest. A more detailed description of dynamic routing problems can be found in [4].

In our problem, the number of packets created by an injector is bounded. Moreover, we have an upper bound for the period of time in which these packets are created and we have a lower bound for the time difference between the creation of two packets.

Definition 3.12 ((α, β, γ)-injector) *Let $\alpha, \beta \in \mathbb{N}_0$, $\gamma \in \mathbb{N}$. An injector I_i, $i \in [m]$ is an (α, β, γ)-injector if and only if the following conditions are fulfilled:*

1. *I_i creates α packets.*
2. *Every created packet has destination $m - 1$.*
3. *The last packet is created after at most β steps.*

4. *If two packets are created in steps t_1 and t_2, then $|t_1 - t_2| \geq \gamma$. We call γ the creation time of the injector.*

□

An (α, β, γ)-injector I_i works in the following way. A created packet is inserted into node i in step t if I_i is *enabled* in step t and no packet enters node i in step t. The packets are created in the beginning of a step. Hence a packet created in step t by I_i can be inserted into node i in step t and can be sent to node $i+1$ in step t.

It is very difficult to analyse such situations in general. We restrict us to cases needed in Chapter 4. We do not allow that a packet can be created at any time and we assume that the creation rate of an injector depends on the number of packets created by the injector. Furthermore, we do not allow that two or more created packets are in an injector in the same step. This leads to the following two definitions.

Definition 3.13 *An (α, β, γ)-injector is called a restricted (α, β, γ)-injector with offset x, if the following conditions are fulfilled:*

- *If a packet is created in step t, then $t \equiv x \pmod{\gamma}$.*
- *If a created packet is not inserted at most $\gamma - 1$ steps after its creation, then it is deleted by the injector.*

Definition 3.14 *Let $k, T, T' \in \mathbb{N}, T' \leq T$. For $i \in [m]$ let I_i be an $(\alpha_i, \beta_i, \gamma_i)$ injector. We call the set $\mathcal{I} = \{I_0, I_1, \ldots, I_{m-1}\}$ a set of (k, T', T) bounded injectors if the following conditions are fulfilled:*

- *$\forall i \in [m] : I_i$ is a restricted $(\alpha_i, \beta_i, \gamma_i)$-injector with offset i, where*

- $\beta_i = T$,
- $\alpha_i > 0 \Longrightarrow \gamma_i = \frac{T'}{\langle \alpha_i \rangle} \in \mathbb{N}$, *where*

$$\begin{array}{rccl} \langle\rangle : & \mathbb{N}_0 & \longrightarrow & \mathbb{N}_0 \\ & x & \mapsto & \langle x \rangle = \begin{cases} 2^l & : \quad \exists l \in \mathbb{N}_0 : 2^l \leq x < 2^{l+1} \\ 0 & : \quad \text{else} \end{cases} \end{array}$$

- $\sum_{i \in [m]} \alpha_i \leq k$,

Now we can present the problem considered in this section.

Problem 2: Dynamic Routing

Given is a one-dimensional mesh of size m, where every node $i \in [m]$ has an injector I_i and the set $\{I_0, \ldots, I_{m-1}\}$ is a set of (k, T', T) bounded injectors. In the beginning there is no packet on a node. The injectors have some additional information (which is described later). The following two tasks have to be fulfilled:

P2.1 Route all created packets to node $m - 1$.

P2.2 The maximal number of packets on any node in any step is 1.

□

For the rest of this section we assume that I_i, $i \in [m]$ is an $(\alpha_i, \beta_i, \gamma_i)$ injector and $\{I_0, \ldots, I_{m-1}\}$ is a set of (k, T', T) bounded injectors for some $k, T', T \in \mathbb{N}$.

We route the packets in the one-dimensional mesh on the shortest path from their source node (the node into which they are injected) to node $m - 1$. After insertion, the packets travel to $m-1$ without delay, i.e., a packet that reaches node $i < m$ in step t is sent to node $i + 1$ in the same step and reaches node $i + 1$ in step $t + 1$. Hence **P2.2** is fulfilled.

It is a little bit more complicated to fulfill **P2.1**. The above routing strategy transports the packets like a conveyor belt. It moves packets one node per step. In the following, the packets are transported within boxes on a conveyor belt. The boxes are moved from node 0 to node m. Each box consists of several slots. In each of these slots at most one packet can be transported. An enabled injector can insert a created packet into a slot if there is no packet in the slot. We call boxes *intervals* and slots steps. The number of steps of an interval depends on the maximal number of packets created by an injector. Let $\bar{\imath} \in [m]$ be any index such that $\alpha_{\bar{\imath}} = \max_{i \in [m]} \alpha_i$.

Definition 3.15 (l-th interval, x-th step, length) *Let $i \in [m]$, $\alpha_i > 0$, $l \in \mathbb{Z}$. The l-th interval of I_i is the interval $[l\gamma_{\bar{\imath}} + i, (l+1)\gamma_{\bar{\imath}} + i)$ and is denoted by $T_{l,i}$. For $1 \leq x \leq \gamma_{\bar{\imath}}$ we call $l\gamma_{\bar{\imath}} + i + x - 1$ the x-th step of the l-th interval of I_i. The length of an interval is $\gamma_{\bar{\imath}}$.*

The next lemma follows directly from the above definition.

Lemma 3.16 *Let $s(p) \in [m]$ be the source of packet p, i.e., p is inserted by injector $I_{s(p)}$. All packets p travel from node $s(p)$ to node $m-1$ without delay, then the following holds: $\forall i \in [m], \forall l \in \mathbb{Z}, \forall x$, $1 \leq x \leq \gamma_{\bar{\imath}}$:*

If p is inserted by I_i in the x-th step of the l-th interval of I_i, then p reaches node i', $i' \in \{s(p), \ldots, m-1\}$, in the x-th step of the l-th interval of $I_{i'}$.

The injector $I_{\bar{\imath}}$ creates a maximal number of packets and has a minimal creation time. $I_{\bar{\imath}}$ can create one packet in each interval. For $i \in [m], \alpha_i > 0$, we set $\delta_i \stackrel{\text{def}}{=} \frac{\gamma_i}{\gamma_{\bar{\imath}}} = \frac{\langle \alpha_{\bar{\imath}} \rangle}{\langle \alpha_i \rangle}$. Injector I_i creates at most one packet during δ_i intervals. If I_i creates a packet in step t, then $t = l\delta_i\gamma_{\bar{\imath}} + i$ for a $l \in \mathbb{Z}$, i.e., the packet is created in the first step of $T_{l\delta_i,i}$. A packet created by injector I_i in the first step of $T_{l\delta_i,i}$ will be deleted by I_i at the end of the γ_i-th step of $T_{(l+1)\delta_i-1,i}$, provided it is in the injector at the end of this step.

In the case an injector I_i inserts a created packet into node i in the first step without a packet on node i, it could happen that long continuous sequences of packets are built. These sequences prevent other injectors from injecting their packets. We give an example for such a situation:

Example:
Assume that all injectors have a creation time of $x < m$. An injector I_i, where $i = rx + s,\ r, s \in \mathbb{N}_0,\ 0 \leq s < x$, is able to create a packet in steps t, where $t \equiv s \pmod{x}$. If injector I_i, $i \in [x]$ creates a packet in step i and inserts it into node i in step i, then a sequence of x packets is built that passes node x in steps $x, \ldots, 2x - 1$. If I_x creates a packet in step x, then this packet has to be deleted by I_x in step $2x - 1$. $\diamond$

We restrict the period of time in which the injectors are enabled to avoid such situations. The purpose of this restriction is to bound the maximal number of packets in a node during any period of time. We begin with the following definition.

Definition 3.17 *For all $j \in \mathbb{N}_0$ we define $U_j \stackrel{\text{def}}{=} \{i \in [m] \mid \langle \alpha_i \rangle = 2^j\}$. We call an injector I_i, $i \in U_j$ an injector of U_j. We further define*

- $\bar{\jmath} \stackrel{\text{def}}{=} \max\{l \in \mathbb{N}_0 \mid U_l \neq \emptyset\}$,
- $\forall j \in [\bar{\jmath} + 1] : \theta_j \stackrel{\text{def}}{=} 2^{\bar{\jmath} - j}$, *and*
- $\forall j \in [\bar{\jmath} + 1] : s_j \stackrel{\text{def}}{=} \left\lfloor \frac{|U_j|}{\theta_j} \right\rfloor$ *and* $r_j \stackrel{\text{def}}{=} |U_j| \mod \theta_j$.

Note that $\bar{\imath} \in U_{\bar{\jmath}}$. The injectors of $U_{\bar{\jmath}-j}$, $j \in [\bar{\jmath}+1]$ create at most one packet in $2^{\bar{\jmath}-j} = \theta_j$ consecutive intervals. Furthermore, for all injectors I_i of U_j, we have $\delta_i = \theta_j$.

For all injectors I_i of U_j, we calculate a value $\eta_i \in [\theta_j]$ and enable injector I_i in intervals $T_{l,i}$ such that $l \equiv \eta_i \pmod{\theta_j}$ and disable I_i in all other intervals.

Definition 3.18 *We say an injector I_i, $i \in [m]$, $\alpha_i > 0$, is enabled in interval $l \in \mathbb{Z}$ if and only if $l \equiv \eta_i \pmod{\delta_i}$.*

Our purpose is to bound the number of injectors that are enabled in an interval l.

For all $j \in [\bar{\jmath}+1]$, we number the injectors of U_j from 0 to $|U_j|-1$. For an injector I_i of U_j with number x, we set $\eta_i \stackrel{\text{def}}{=} x \pmod{\delta_i}$.

Lemma 3.19 $\forall j \in [\bar{\jmath}+1]\, \forall l \in \mathbb{Z}$: *At most s_j+1 injectors of U_j are enabled in interval l, i.e. $s_j + 1 \geq |\{i \in [m] \mid \alpha_i > 0,$ injector I_i is an injector of U_j, $\eta_i \equiv l \pmod{\delta_i}\}|$.*

Proof:
There are $s_j\theta_j + r_j$ injectors of U_j. For all injectors $i \in U_j$, we have $\delta_i = \theta_j$. Hence at most $\left\lceil \frac{|U_j|}{\delta_i} \right\rceil \leq s_j+1$ injectors of U_j are enabled in an interval $l \in \mathbb{Z}$.
$\diamond$

The following lemma is a simple consequence of Lemma 3.19.

Lemma 3.20 $\forall l \in \mathbb{Z}$: *At most $\sum_{j\in[\bar{\jmath}+1]}(1 + s_j)$ injectors are enabled in interval l.*

If T' is large enough, then **P2.1** can be fulfilled.

Lemma 3.21 *If $T' \geq 2^{\bar{\jmath}} \sum_{j\in[\bar{\jmath}+1]}(1+s_j)$, all packets travel from their source node to $m-1$ without delay, and each injector I_i, $\alpha_i > 0$, $i \in [m]$ inserts its created packets only in intervals $T_{l,i}$ such that $l \equiv \eta_i \pmod{\delta_i}$, then each injector is able to insert all of its created packets.*

Proof:
An injector is able to insert at most one packet into a node during an interval.

Due to Lemma 3.20 at most $\sum_{j\in[\bar{\jmath}+1]}(1+s_j)$ injectors are enabled in an interval.

An interval has length $\gamma_{\bar{\imath}}$. It holds

$$\begin{aligned}\gamma_{\bar{\imath}} &= \frac{T'}{\langle\alpha_{\bar{\imath}}\rangle}\\ &= \frac{T'}{2^{\bar{\jmath}}}\\ &\geq \frac{2^{\bar{\jmath}}\sum_{j\in[\bar{\jmath}+1]}(1+s_j)}{2^{\bar{\jmath}}}\\ &= \sum_{j\in[\bar{\jmath}+1]}(1+s_j).\end{aligned}$$

Thus each injector is able insert all its created packets. ◇

Now we can give an upper bound for the time needed to solve an instance of **Problem 2**.

Theorem 3.22 *If all injectors $I_i, \alpha_i > 0, i \in [m]$ know η_i, δ_i and $\gamma_{\bar{\imath}}$, $T' \geq 2^{\bar{\jmath}}\sum_{j\in[\bar{\jmath}+1]}(1+s_j)$, and all packets travel from their source node to node $m-1$ without delay, then any such instance of* **Problem 2** *can be solved in $T+T'+m$ steps. $O(\log T')$ bits are sufficient for each injector to enable and disable it.*

Proof:

Due to Lemma 3.20 every created packet can be inserted before it is deleted. After T steps all injectors have created their packets. After at most $T+T'$ steps all packets are inserted. After insertion a packet has to travel at most m steps. Hence the last packet reaches its destination after at most $T+T'+m$ steps.

To enable and disable the injectors at the right times we have to store $\gamma_{\bar{\imath}}$, δ_i and η_i and we have to count steps in an interval and intervals. We need to count up to $\max\{\delta_i \mid i \in [m], \alpha_i > 0\}$ intervals. An interval consists of $\gamma_{\bar{\imath}}$

steps. For all $i \in [m], \alpha_i > 0$, we have $\eta_i \leq \delta_i \leq \gamma_i \leq T'$. Hence $O(\log T')$ bits are sufficient to enable and disable an injector. $\diamond$

Corollary 3.23 *If all injectors $I_i, \alpha_i > 0, i \in [m]$ know η_i, δ_i and $\gamma_{\bar{\imath}}$, $T' \geq 2^{\bar{\jmath}} \sum_{j \in [\bar{\jmath}+1]}(1 + s_j)$, and all packets travel from their source node to node $m-1$ without delay, then any such instance of* **Problem 2** *can be solved in $2T + m$ steps.*

We have shown an upper bound of $2T + m$ steps for the case $T' \geq \bar{\jmath}2^{\bar{\jmath}} + 2^{\bar{\jmath}} \sum_{j \in [\bar{\jmath}+1]} s_j$. Unluckily $\bar{\jmath}2^{\bar{\jmath}} + 2^{\bar{\jmath}} \sum_{j \in [\bar{\jmath}+1]} s_j$ can be very large. There exist instances of **Problem 2**, where $\bar{\jmath} = \lfloor \log_2 k \rfloor$[1], for example in the case $\alpha_i = k$ for a $i \in [m]$. In this case

$$\bar{\jmath}2^{\bar{\jmath}} + 2^{\bar{\jmath}} \sum_{j \in [\bar{\jmath}+1]} s_j \geq \langle k \rangle \lfloor \log_2 k \rfloor.$$

We now improve the condition on T' to $T' \geq 2^{\bar{\jmath}} + k$. We begin with an extended version of Definition 3.17.

Definition 3.24 *For all $j \in \mathbb{N}_0$ we define $U_j \stackrel{\text{def}}{=} \{i \in [m] \mid \langle \alpha_i \rangle = 2^j\}$. We call an injector I_i, $i \in U_j$ an injector of U_j. We further define*

- $\bar{\jmath} \stackrel{\text{def}}{=} \max\{l \in \mathbb{N}_0 \mid U_l \neq \emptyset\}$,
- $\forall j \in [\bar{\jmath}+1] : \theta_j \stackrel{\text{def}}{=} 2^{\bar{\jmath}-j}$,
- $\forall j \in [\bar{\jmath}+1] : s_j \stackrel{\text{def}}{=} \left\lfloor \frac{|U_j|}{\theta_j} \right\rfloor$, $r_j \stackrel{\text{def}}{=} |U_j| \mod \theta_j$, *and*
- $\forall j \in [\bar{\jmath}+1] : R_j \stackrel{\text{def}}{=} \sum_{i=j}^{\bar{\jmath}} r_i 2^{i-j}$, $x_j \stackrel{\text{def}}{=} \left\lfloor \frac{R_j}{\theta_j} \right\rfloor$, $y_j \stackrel{\text{def}}{=} R_j \mod \theta_j$.

For all $j \in [\bar{\jmath}+1]$ we number the injectors of U_j from 0 to $|U_j| - 1$. An injector of U_j with a number from 0 to $s_j\theta_j - 1$ is called *normal* and an injector of U_j with a number from $s_j\theta_j$ to $|U_j| - 1$ is called *special* . For a normal injector I_i of U_j with number $x \in [s_j\theta_j]$ we set $\eta_i \stackrel{\text{def}}{=} x \pmod{\delta_i}$.

[1] logarithm base 2

Lemma 3.25 $\forall j \in [\bar{\jmath}+1]\,\forall l \in \mathbb{Z}$: *At most* s_j *normal injectors of* U_j *are enabled in interval* l*, i.e.* $s_j \leq |\{i \in [m] \mid \alpha_i > 0,$ *injector* I_i *is a normal injector of* U_j, $\eta_i \equiv l \pmod{\delta_i}\}|$.

Proof:

The proof can be done analogously to the proof of Lemma 3.19. There are $s_j\theta_j$ normal injectors of U_j. For all injectors $i \in U_j$ we have $\delta_i = \theta_j$. Hence at most $\frac{s_j\theta_j}{\delta_i} = s_j$ normal injectors of U_j are enabled in an interval $l \in \mathbb{Z}$. $\diamond$

Lemma 3.26 $\forall l \in \mathbb{Z}$: *At most* $\sum_{j\in[\bar{\jmath}+1]} s_j$ *normal injectors are enabled in interval* l.

Calculation of η_i for special injectors. First note that there are no special injectors of $U_{\bar{\jmath}}$. Now we calculate η_i for special injectors of $U_{\bar{\jmath}-1}, \ldots, U_0$. For this purpose we number the special injectors of U_j from 0 to $r_j - 1$ for all $j \in [\bar{\jmath}+1]$.

In the following $b_j \in \{0,1\}^{\theta_j}$ for all $j \in [\bar{\jmath}+1]$. We calculate the b_j inductively beginning with $b_{\bar{\jmath}}$. With their help we calculate η_i. We use the following idea. An injector of U_j creates at most one packet in θ_j consecutive intervals. Each of these θ_j intervals corresponds to one zero or one of b_j. Every special injector of U_j can change exactly one zero in b_j into a one. The position of the zero determines η_i (remember η_i determines the interval in which I_i is enabled). If there are no more zeros in b_j, i.e., $b_j = 1^{\theta_j}$, then b_j is set to 0^{θ_j}.

Calculation of b_j and η_i:

We set $b_{\bar{\jmath}} = 0 \in \{0,1\}^1$. We denote the number of ones in b_j by $\#_1 b_j$ ($\#_0 b_j$ is defined analogously) and write $\circ$ for the concatenation of two words over $\{0,1\}$. If $b_j = a_0 \ldots a_{\theta_j - 1}$, $a_i \in \{0,1\}$, $i \in [\theta_j]$, then a_i is in the i-th position of b_j and is denoted by $(b_j)_i$. If $a_i = 0$ and $k = \#_0 a_0 \ldots a_i$, then i is the position of the k-th zero in $a_0 \ldots a_i$. For $\underbrace{a \ldots a}_{k}$ we write a^k.

injector	I_0	I_1	I_2	I_3	I_4	I_5	$\mathbf{I_6}$	I_7	I_8	I_9	I_{10}	$\mathbf{I_{11}}$	I_{12}	I_{13}	$\mathbf{I_{14}}$	I_{15}
α	1	1	5	6	7	8	17	9	5	6	10	16	6	7	32	18
γ	256	256	64	64	64	32	16	32	64	64	32	16	64	64	8	16
δ	32	32	8	8	8	4	2	4	8	8	4	2	8	8	1	2
U_j	U_0	U_0	U_2	U_2	U_2	U_3	U_4	U_3	U_2	U_2	U_3	U_4	U_2	U_2	U_5	U_4
number	0	1	0	1	2	0	0	1	3	4	2	1	5	6	0	2
special	0	1	0	1	2	0	-	1	3	4	2	-	5	6	-	0
η	1	2	1	2	3	1	**0**	3	5	6	0	**1**	7	0	**0**	0

Table 3.1: An example for the calculation of η_i. (part 1)

For $t \in [r_j]$ let I_{i_t} be the special injector of U_j with number t. Let $j < \bar{\jmath}$:

Case 1: $2\#_0\, b_{j+1} \leq r_j$

We set $b_j = 1^{r_j - 2\#_0\, b_{j+1}} 0^{\theta_j + 2\#_0\, b_{j+1} - r_j}$. If $t < 2\#_0\, b_{j+1}$ and $k_t \in [\theta_j]$ is the position of the $t+1$-th zero in $b_{j+1} \circ b_{j+1}$, then we set $\eta_{i_t} \stackrel{\text{def}}{=} k_t$. If $2\#_0\, b_{j+1} \leq t < r_j$, then we set $\eta_{i_t} \stackrel{\text{def}}{=} t - 2\#_0\, b_{j+1}$.

Case 2: $2\#_0\, b_{j+1} > r_j$

We set $b_j = b_{j+1} \circ b_{j+1}$ and change the first r_j zeros of b_j to one. Let $k_t \in [\theta_j]$ be the position of the $t+1$-th zero in $b_{j+1} \circ b_{j+1}$. We set $\eta_{i_t} \stackrel{\text{def}}{=} k_t$.

Before we begin to proof an upper bound for the number of injectors enabled in an interval we give an example.

Example:

Table 3.1 shows an instance of dynamic routing in a one-dimensional mesh of size 16. We assume that $\{I_0, \ldots, I_{15}\}$ is a set of (k, T', T) bounded injectors where $k = 154$, $T' = 256$, and $T = 512$. For I_0 Table 3.1 provides the following: $\alpha_0 = 1$, $\gamma_0 = 256$, $\delta_0 = 32$, $0 \in U_0$, I_0 is a injector of U_0 with

number 0, I_0 is a special injector of U_0 with number 0, and $\eta_0 = 1$. Injectors I_6, I_{11}, and I_{14} are normal injectors. All other injectors are special injectors. The special injector I_{15} is the only one for which its special number differs from the normal one. I_{14} injects the most packets (32), hence we have $\bar{\imath} = 14$ and $\bar{\jmath} = 5$. We get an interval length of 8. The calculation of η for the special injectors provides $b_5 = 0$, $b_4 = 10$, $b_3 = 1000$, $b_2 = 10000000 = 10^7$, $b_1 = 10^710^7$, and $b_0 = 1110^510^710^710^7$. In the case of I_{15} the value of η_{15} is computed as follows. I_{15} is the only special injector of U_4 hence $r_4 = 1$. Its number (as special injector) is 0. We have $2\#_0\, b_5 = 2$ and $r_4 = 1$. Therefore case 2 is fulfilled. The first zero in $b_5 \circ b_5 = 00$ is in position 0. Hence we get $\eta_{15} = 0$. In the case of I_{13} the value of η_{13} is computed as follows. I_{13} is a special injector of U_2. Its number is 6. We have $2\#_0\, b_3 = 6$. There are seven special injectors of U_2. Thus $r_2 = 7$. So case 1 is fulfilled. We have $2\#_0\, b_3 = 6 < r_2 = 7$. Hence we set $\eta_{13} = 6 - 2\#_0 b_3 = 0$.

At the first sight it surprises that we have five times a value of zero for η and only one time a value of seven.

In Table 3.2.3 we give for all I_i, $i \in [15]$ and for all $l \in [32]$ the intervals $T_{l,i}$ in which I_i is enabled. If injector I_i is enabled in $T_{l,i}$, then we write $\Box$ at the intersection of column I_i and row $T_{l,\cdot}$. The maximal value of δ in our example is 32 and hence we get the same table for $l \in \{32, \ldots, 63\}$, $l \in \{-32, \ldots, -1\}$, $l \in \{64, \ldots, 95\}$, $l \in \{-64, \ldots, -33\}$ and so on. At most five and at least four injectors are enabled in an interval. In the case that five injectors are enable three of them are special injectors. If four injectors are enabled, then two of them are special injectors. Note that five injectors are enabled in $T_{l,\cdot}$ if and only if there is a one in position l of b_0.

$\diamond$

Lemma 3.27 *For all $j \in [\bar{\jmath}]$ let R_j, x_j, and y_j be given as in definition*

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}	I_{11}	I_{12}	I_{13}	I_{14}	I_{15}
$T_{0,\cdot}$							□				□			□	□	□
$T_{1,\cdot}$	□		□			□						□			□	
$T_{2,\cdot}$		□		□			□								□	□
$T_{3,\cdot}$					□			□				□			□	
$T_{4,\cdot}$							□				□				□	□
$T_{5,\cdot}$						□			□			□			□	
$T_{6,\cdot}$							□			□					□	□
$T_{7,\cdot}$								□				□	□		□	
$T_{8,\cdot}$							□				□			□	□	□
$T_{9,\cdot}$			□			□						□			□	
$T_{10,\cdot}$				□			□								□	□
$T_{11,\cdot}$					□			□				□			□	
$T_{12,\cdot}$							□				□				□	□
$T_{13,\cdot}$						□			□			□			□	
$T_{14,\cdot}$							□			□					□	□
$T_{15,\cdot}$								□				□	□		□	
$T_{16,\cdot}$							□				□			□	□	□
$T_{17,\cdot}$			□			□						□			□	
$T_{18,\cdot}$				□			□								□	□
$T_{19,\cdot}$					□			□				□			□	
$T_{20,\cdot}$							□				□				□	□
$T_{21,\cdot}$						□			□			□			□	
$T_{22,\cdot}$							□			□					□	□
$T_{23,\cdot}$								□				□	□		□	
$T_{24,\cdot}$							□				□			□	□	□
$T_{25,\cdot}$			□			□						□			□	
$T_{26,\cdot}$				□			□								□	□
$T_{27,\cdot}$					□			□				□			□	
$T_{28,\cdot}$							□				□				□	□
$T_{29,\cdot}$						□			□			□			□	
$T_{30,\cdot}$							□			□					□	□
$T_{31,\cdot}$								□				□	□		□	

Table 3.2: An example for the calculation of η_i. (part 2)

3.24. The following holds:

1. $R_j = r_j + x_{j+1}\theta_j + 2y_{j+1}$
2. $x_j \in \{x_{j+1}, x_{j+1}+1\}$
3. $(x_j = x_{j+1}) \Longleftrightarrow (r_j + 2y_{j+1} < \theta_j)$
4. $y_j = r_j + 2y_{j+1} \bmod \theta_j$

Proof:

1. $$\begin{aligned} R_j &= \sum_{i=j}^{\bar{j}} r_i 2^{i-j} \\ &= r_j + 2R_{j+1} \\ &= r_j + 2x_{j+1}\theta_{j+1} + 2y_{j+1} \\ &= r_j + x_{j+1}\theta_j + 2y_{j+1}. \end{aligned}$$
2. $$\begin{aligned} x_j &= \left\lfloor \frac{R_j}{\theta_j} \right\rfloor \\ &= x_{j+1} + \left\lfloor \frac{r_j + 2y_{j+1}}{\theta_j} \right\rfloor. \end{aligned}$$

   The result follows directly from the fact that $0 \leq r_j + 2y_{j+1} < 2\theta_j$.
3. Follows from the proof of 2.
4. $$\begin{aligned} y_j &= R_j \bmod \theta_j \\ &= r_j + x_{j+1}\theta_j + 2y_{j+1} \bmod \theta_j \\ &= r_j + 2y_{j+1} \bmod \theta_j \end{aligned}$$

◇

Lemma 3.28 $\forall l \in \mathbb{Z}$: *At most* $\frac{\sum_{i=0}^{\bar{j}} r_i 2^i}{2^j} + 1$ *special injectors are enabled in interval* l*.*

Proof:

First note that for all $j \in [\bar{j}+1]$ at most two special injectors of U_j are enabled in l because there are $r_j < \theta_j$ special injectors of U_j, i.e., only $r_i \leq \theta_j - 1$ zeros of b_j are changed into ones.

We prove the following statements by induction on j:

- $\forall j \in [\bar{\jmath}+1] : y_j = \#_1\, b_j$
- $\forall j \in [\bar{\jmath}+1]\, \forall l \in \mathbb{Z}$: At most z_j special injectors of $U_{\bar{\jmath}} \cup U_{\bar{\jmath}-1} \cup \cdots \cup U_j$ are enabled in interval l, where

$$z_j = \begin{cases} x_j & : \quad (b_j)_l = 0 \\ x_j + 1 & : \quad (b_j)_l = 1 \end{cases}$$

Here we write $(b_j)_l$ for $(b_j)_{l \bmod \theta_j}$.

The statements are fulfilled for $j = \bar{\jmath}$. If $r_{\bar{\jmath}-1} = 0$, then $x_{\bar{\jmath}-1} = y_{\bar{\jmath}-1} = 0$ and $b_{\bar{\jmath}-1} = 00$. If $r_{\bar{\jmath}-1} = 1$, then $x_{\bar{\jmath}-1} = 0$, $y_{\bar{\jmath}-1} = 1$ and $b_{\bar{\jmath}-1} = 10$. Hence the statements are fulfilled for $j = \bar{\jmath} - 1$.

Now let $0 \leq j < \bar{\jmath} - 1$. We distinguish two cases.

Case $0 \leq 2\#_0\, b_{j+1} \leq r_j$**:**

Note that $0 \leq \#_1\, b_j < \theta_j$. Using Lemma 3.27 and induction hypothesis we get

$$\begin{aligned} \#_1\, b_j &= r_j - 2\#_0\, b_{j+1} && \text{(case 1, page 37)} \\ &= r_j - 2(\theta_{j+1} - \#_1\, b_{j+1}) \\ &= r_j - \theta_j + 2\#_1\, b_{j+1} && \text{(Def. } \theta_j) \\ &= r_j - \theta_j + 2y_{j+1} && \text{(ind. hyp.)} \\ &= y_j && \text{(Lemma 3.27)} \end{aligned}$$

We have $r_j - \theta_j + 2y_{j+1} \geq 0$. So we get $r_j + 2y_{j+1} \geq \theta_j$. Lemma 3.27 provides $x_j = x_{j+1} + 1$. We have four subcases.

Subcase $(b_{j+1})_l = 0 \wedge (b_j)_l = 1$**:**

Two special injectors of U_j are enabled in l. One special injector of U_j with a number $\leq \#_0\, b_{j+1}$ and one special injector of U_j with a number $> \#_0\, b_{j+1}$. We have at most z_{j+1} enabled injectors of $U_{\bar{\jmath}} \cup \cdots \cup U_{j+1}$. Hence there are at most $z_{j+1} + 2$ enabled injectors of $U_{\bar{\jmath}} \cup \cdots \cup U_j$. It is $(b_j)_l = 1$. Hence by definition $z_j = x_j + 1$. Thus we have have to prove that $z_{j+1} + 2 \leq x_j + 1$.

$$\begin{aligned} z_{j+1} + 2 = &\quad x_{j+1} + 2 \qquad (\ (b_{j+1})_l = 0) \\ = &\quad x_j + 1 \qquad (\ x_j = x_{j+1} + 1) \end{aligned}$$

Subcase $(b_{j+1})_l = 0 \wedge (b_j)_l = 0$**:**

We have $z_j = x_j$. One special injector of U_j is enabled in l (a special injector with a number $\leq \#_0 b_{j+1}$). We have $z_{j+1} + 1 = x_{j+1} + 1 = x_j$.

Subcase $(b_{j+1})_l = 1 \wedge (b_j)_l = 0$**:**

We have $z_j = x_j$. No special injector of U_j is enabled. We have $z_{j+1} = x_{j+1} = x_j - 1 \leq x_j$.

Subcase $(b_{j+1})_l = 1 \wedge (b_j)_l = 1$**:**

We have $z_j = x_j + 1$. One special injector of U_j is enabled in l. We have $z_{j+1} + 1 = x_{j+1} + 1 = x_j \leq x_j + 1$.

Case $2 \cdot \#_1\ b_{j+1} > r_j$**:**

Using Lemma 3.27 and induction hypothesis we get

$$\begin{aligned} \#_1\ b_j &= r_j + 2\#_1\ b_{j+1} \\ &= r_j + 2y_{j+1} \\ &= y_j \end{aligned}$$

We get $r_j + 2y_{j+1} < \theta_j$ and hence $x_j = x_{j+1}$. Note that in this case maximal one special injector of U_j is enabled. Three subcases are possible.

Subcase $(b_{j+1})_l = 0 \wedge (b_j)_l = 1$**:**

One special injector of U_j is enabled in l. We have $z_{j+1} + 1 = x_{j+1} + 1 = x_j + 1$.

Subcase $(b_{j+1})_l = 0 \wedge (b_j)_l = 0$**:**

No special injector of U_j is enabled in l. We have $z_{j+1} = x_{j+1} = x_j$.

Subcase $(b_{j+1})_l = 1 \wedge (b_j)_l = 1$**:**

No special injector of U_j is enabled in l. We have $z_{j+1} = x_{j+1} = x_j$.

By Lemma 3.27 we get $x_{\bar{\jmath}} \leq x_{\bar{\jmath}-1} \leq \cdots \leq x_0$. We have $x_0 + 1 \leq \frac{\sum_{i=0}^{\bar{\jmath}} i2^i}{2^{\bar{\jmath}}} + 1$ (see Definition 3.24). ◇

Now we can bound the number of enabled injectors.

Lemma 3.29 $\forall l \in \mathbb{Z}$: *At most* $\frac{k}{2^{\bar{\jmath}}} + 1$ *injectors are enabled in interval* l.

Proof:

By Lemma 3.26 at most $\sum_{i=0}^{\bar{\jmath}} s_i$ normal injectors are enabled in l. By Lemma 3.28 at most $\frac{\sum_{i=0}^{\bar{\jmath}} r_i 2^i}{2^{\bar{\jmath}}} + 1$ special injectors are enabled in l. Remember that $s_j = \frac{|U_j| - r_j}{\theta_j}$ and note that $k \geq \sum_{i \in [\bar{\jmath}+1]} 2^i |U_i|$. We get:

$$\begin{aligned} \frac{\sum_{i=0}^{\bar{\jmath}} r_i 2^i}{2^{\bar{\jmath}}} + \sum_{i=0}^{\bar{\jmath}} s_i + 1 &= \frac{\sum_{i=0}^{\bar{\jmath}} 2^i |U_i|}{2^{\bar{\jmath}}} + 1 \\ &\leq \frac{k}{2^{\bar{\jmath}}} + 1 \end{aligned}$$

◇

This leads to an improved version of Lemma 3.21.

Lemma 3.30 *If* $T' \geq k + 2^{\bar{\jmath}}$, *all packets travel from their source to node* $m - 1$ *without delay and each injector* I_i, $\alpha_i > 0$, $i \in [m]$ *inserts its packets only in intervals* $T_{l,i}$ *such that* $l \equiv \eta_i \pmod{\delta_i}$, *then each injector is able to insert all of its packets.*

Proof:

The proof can be done analogously to the proof of Lemma 3.21.

◇

This yields

Theorem 3.31 *If all injectors $I_i, \alpha_i > 0, i \in [m]$ know η_i, δ_i and $\gamma_{\bar{i}}$, $T' \geq k + 2^{\bar{j}}$ and all packets travel from their source node to $m-1$ without delay, then any such instance of* **Problem 2** *can be solved in $T + T' + m$ steps. $O(\log T')$ bits are needed for each injector to enable and disable it.*

Proof:
The proof can be done analogously to the proof of Theorem 3.22.

◇

In the following corollary we simplify the condition on T' a little bit.

Corollary 3.32 *If all injectors $I_i, \alpha_i > 0, i \in [m]$ know η_i, δ_i and $\gamma_{\bar{i}}$, $T' \geq 2k$ and all packets travel from their source node to $m-1$ without delay, then any such instance of* **Problem 2** *can be solved in at most $2T + m$ steps. $O(\log T')$ bits are sufficient for each injector to enable and disable it.*

Proof:
Observe that $2^{\bar{j}} \leq k$ and $T' \leq T$. ◇

In a k'-k' routing problem (on a higher dimensional mesh) a one-dimensional (sub)mesh of size m is destination of at most $k'm$ packets. Now assume that these packets are inserted by a set of $(k'm, T, T')$ bounded injectors, where $T \in O(k'm)$, and $T' \geq 2k'm$, then we are able to route the packets to their destination in $O(k'm)$ steps with buffer size 1, provided every node i knows η_i, δ_i and $\gamma_{\bar{i}}$.

Corollary 3.33 *Let $k = k'm$, $T' \geq 2k'm$, $T \in O(k'm)$. If all injectors $I_i, \alpha_i > 0, i \in [m]$ know η_i, δ_i and $\gamma_{\bar{i}}$, all packets travel from their source node to $m-1$ without delay, then any such instance of* **Problem 2** *can be solved in $O(k'm)$ steps.*

3.3 Conclusion.

We summarize the results of this section. We have achieved the following:

embedding: All presented algorithms work on networks of fixed degree with a constant slowdown and at most constant additional buffer size.

gap routing: On a one-dimensional mesh of size m, we give an algorithm that produces a stream of packets such that two packets of the same class reach their destination with a given time difference. In the case that at most k packets are stored on a processor initially and an upper bound of m' for the product of the time difference and the number of packets in a class exists, the last packet reaches its destination after at most $m + km + m' \in O(km + m')$ steps. The algorithm routes the packets on a shortest path to their destination, is oblivious, and has a buffer size of $k + 1$.

dynamic routing: We are able to solve special dynamic routing problems. In the case that at most $O(km)$ packets are created and some additional requirements are fulfilled, we can give an algorithm that solve the problem in $O(km)$ steps, routes the packets on a shortest path to their destination, is oblivious, and has a buffer size of one.

Chapter 4

Oblivious Routing.

This chapter considers k-k routing problems. We give an oblivious k-k routing algorithm with running time $O(k|V_{\mathcal{N}}|^{\frac{1}{2}})$ and buffer size $O(k)$ for a network $\mathcal{N}$ for which a partitioning into blocks exists. Our main interest lies in the design of oblivious k-k routing algorithms for r-dimensional meshes of side length n. We achieve asymptotically optimal running times for these networks.

4.1 Introduction.

The concept of oblivious routing strategies was introduced by Valiant in [49]. A routing algorithm is called oblivious if the path of each packet only depends on its source and destination node and is completely independent of the paths of all other packets (for a formal definition see Definition 2.1 on page 13).

Therefore, the path on which a packet is routed to its destination can be determined before the routing process starts. Furthermore, if all paths used by packets are simple paths, it is sufficient that a processor maintains a table

with one entry for each possible source and destination pair of a packet to decide on which link a packet leaves the processor. Whenever such a table exists for every processor, no calculations are needed to determine the next link of a packet. In the case that the paths are not simple, more than one entry for every source and destination pair is needed. Such tables can be very large but often in structures as meshes similar source and destination pairs use similar edges and hence the size of the table can be reduced at the cost of some additional calculations.

Nevertheless, the restriction of allowed paths makes oblivious routing simple and hence attractive. Furthermore, it is of theoretical interest how fast it is possible to route packets under such restrictions. Hence oblivious routing was considered in several publications, e.g. [2, 23, 41, 29, 16, 43, 3, 37, 40, 32, 9, 14, 13]. It was shown that the simplicity has its costs in the running time. Borodin and Hopcraft [2] have shown an $\Omega(\frac{\sqrt{|V_{\mathcal{N}}|}}{deg(\mathcal{N})^{3/2}})$ lower bound for oblivious permutation routing on a network $\mathcal{N}$. In [16], Kaklamanis, Krizanc, and Tsantilas improved this bound:

Theorem 4.1 (Theorem 6 in [16].) *In any network with N nodes and degree d, any oblivious k-k routing algorithm requires $\Omega(k\sqrt{N}/d+k/d)$ steps in the worst case.*

Since the family of r-dimensional meshes is of fixed degree, the above theorem results in a $\Omega(kn^{\frac{r}{2}})$ lower bound for oblivious k-k routing on r-dimensional meshes with side length n.

Furthermore, Krizanc [22] has shown that any *pure* and oblivious permutation routing algorithm with buffer size $O(1)$ on a network $\mathcal{N}$ needs at least $\Omega(|V_{\mathcal{N}}|)$ steps. In a pure routing algorithm a packet have to move if the next link of its path is not used by another packet.

In a network having a Hamiltonian path a trivial upper bound for the run-

ning time of a pure and oblivious k-k routing algorithm with buffer size $O(k)$ is $O(k|V_{\mathcal{N}}|)$. In r-dimensional meshes a Hamiltonian path exists. For these networks it was unknown for a long time whether a deterministic, oblivious, and non pure k-k routing algorithm with buffer size $O(k)$ exists that beat the trivial upper bound.

In [8, 9], the first oblivious permutation routing algorithms for r-dimensional meshes with $O(1)$ buffer size were presented that beat the bound. The algorithm for a two-dimensional mesh of side length n has running time $O(n^{3/2})$ and buffer size $O(1)$. The key observation in [8, 9] was that it is possible to sort packets during oblivious routing. Furthermore, in [8, 9], a technique to obtain oblivious permutation routing algorithms for higher dimensional meshes using an algorithm for a two-dimensional mesh was presented. With the help of this technique oblivious permutation routing algorithms with running time $O(n^{(2r-1)/2})$ and buffer size $O(1)$ were achieved for $M_{r,n}$, $r \geq 2$. In two subsequent papers, the results for the two-dimensional case were improved. In [11, 14][1] an oblivious permutation routing algorithm with running time $O(n)$ and buffer size two[2] was presented. In [12][3], the constant for the running time of the algorithm was reduced at the cost of increasing the constant for the buffer size. The permutation routing algorithm presented in [12] has running time $(2.954+\frac{8\sqrt{d}+8}{d}+2c)n$ and buffer size $2d+16\sqrt{d}+16+\frac{2}{c}$[4]

[1] The proof for the running time of $O(n)$ given in the paper is based on a statement which can shown to be wrong. A similar statement is also used in the proof of the running time in [12]. We do not know whether this problem can be fixed or not, i.e., we are not able to show that the algorithm uses $O(n)$ steps and we are not able to construct a permutation for which the algorithm uses $\omega(n)$ steps. We discuss the problem in Section 4.2 in more detail.

[2] In the model in [11, 14] each link has an input and an output buffer of size 2. Using our model the algorithm need a buffer size of sixteen.

[3] See footnote to [11, 14].

[4] This yields a buffer size of $8(2d + 16\sqrt{d} + 16 + \frac{2}{c})$ in our model.

for constants c and d such that $c < \frac{1}{2}$ and $d = 2^{2l}$ for an integer l.

An application of the techniques of [8, 9] and [12] results in an oblivious permutation routing algorithm with running time $O(n^{r-1})$ and buffer size $O(1)$ for $M_{r,n}$, $r \geq 2$.

In [40], we improved these results. For $M_{r,n}$, $r \geq 2$, an oblivious permutation routing algorithm with running time $O(n^{r/2} \log n)$ and buffer size $O(1)$ was presented. A partitioning of the meshes into blocks and sorting of blocks was used to achieve the result.

In [32], a new model for oblivious routing, the *relaxed model*, was presented. In this model processors can freely send data to their neighbours, i.e., data can be sent that is not accomplished to packets. Hence the model is not a store-and-forward packet routing model. Nevertheless, the lower bound of Theorem 4.1 holds. For both models, oblivious permutation routing algorithms with a running time of $O(n)$[5] and buffer size $O(1)$ are presented for the two-dimensional mesh. The algorithms transport the packets on shortest paths from source to destination. In the algorithm for the relaxed model, all packets take a *strongly-dimensional* path. In an r-dimensional mesh a strongly-dimensional path $p_0 p_1 \cdots p_{r-1}$ is a shortest path between two nodes, where each p_i is a simple path that uses only edges of the i-th dimension. In the algorithm for the standard model, all packets take a *weakly-dimensional* path. In a r-dimensional mesh a weakly-dimensional path $p_{\pi(1)} p_{\pi(2)} \cdots p_{\pi(r)}$ is a shortest path between two nodes, where $\pi \in S_r$ and each $p_{\pi(i)}$ is a simple path that uses only edges of the $\pi(i)$-th dimension. In [40, 10, 13], weakly-dimensional paths are called *elementary*. There it was shown that

[5] A proof of the running time for one part (vertical routing) of their algorithm is missing. Even a proof idea is not given. We are able to construct a very simple instance for which the presented vertical routing algorithm uses $\omega(n)$ steps. We discuss the details of this construction in Section 4.2.

for $r \geq 3$, r an odd integer, it is impossible to achieve the bound of Theorem 4.1 in $M_{r,n}$ using oblivious permutation routing algorithms where all packets use elementary paths. More precisely, in [10, 13] an $\Omega(n^2)$ lower bound for $M_{3,n}$ and in [40] an $\Omega(n^{(r+1)/2})$ lower bound for $M_{r,n}$, r an odd integer, for oblivious permutation algorithms where all packets use elementary paths was given. Therefore, to achieve the bound of Theorem 4.1 for all r on $M_{r,n}$, we are not able to use elementary paths. The paths used by our algorithm are not simple and hence they are not elementary. Furthermore, we do not use sorting to achieve our result.

The rest of this chapter is organized as follows. In the next section we discuss aspects of the oblivious permutation routing algorithms for the two-dimensional mesh presented in [11, 14, 12, 32]. We point out some gaps in the proof or design of these algorithms. In Section 4.3 we give a high level description of our algorithm. In Section 4.4 we present an oblivious k-k routing algorithm for a network $\mathcal{N}$ and r-dimensional meshes an in Section 4.5 we give a conclusion and suggestions for further work.

4.2 Discussion of Previous Results.

4.2.1 The Algorithm Presented in [11, 14].

In [8, 9], the first oblivious permutation routing algorithm for meshes with buffer size $O(1)$ was presented that beat the bound of Theorem 4.1. In [11, 14] a refined version of this algorithm was given and the first oblivious permutation algorithm with an asymptotically optimal running time and buffer size $O(1)$ for the two-dimensional mesh was achived.

The proof of the running time in [11, 14] uses a statement for which we are able to construct an counterexample. We do not know whether this problem

can be fixed or not, i.e., we are not able to show that the algorithm needs $O(n)$ steps and we are not able to construct a permutation for which the algorithm needs $\omega(n)$ steps. Here, we do not discuss the whole algorithm. We restrict our attention to the problem, i.e., we give an example for which the statement does not hold. For more details of the algorithm we refer the reader to the original work.

The problem occurs in stage 3 of the algorithm. This stage is also called critical zone. In the following, we describe the problem that is solved in the critical zone, give the statement used in the proof, and construct a counterexample. The notation we use to describe the problem differs from the notation used in [11, 14].

Description of the Problem.

Given is a two-dimensional mesh of side length $2n$, $n \in \mathbb{N}$. Every node in the set $S \stackrel{\text{def}}{=} \{(x, y) \mid x, y \in [n]\}$ is source of at most one packet. Every node in the set $D \stackrel{\text{def}}{=} \{(x, y) \mid x, y \in [2n] - [n]\}$ is destination of at most one packet. Let $k_{i,j}$ denote the number of packets in row $i \in [n]$ with destination in column $j \in [2n] - [n]$.

($\star$) For all $j \in [2n] - [n]$, $i \in [n]$ and for any two packets p_1, p_2 initially on node $(i, x_1), (i, x_2)$, $x_1, x_2 \in [n], x_1 \neq x_2$ with destination column j, $|x_1 - x_2| \geq \left\lceil \frac{n}{2k_{i,j}} \right\rceil$ holds.

The packets first travel within the row to their destination column and enter (if possible) the destination column. Then they travel within the column to their destination node. The packets in column $j \in [n]$ begin to travel to their destination column in step $4(n - j - 1)$. They travel without being delayed until they reach their destination column. Hence a packet initially in column j with destination column j' reaches its destination column in

step $4(n-j-1)+j'-j$.

The Statement and a Counterexample.

In the following let $j \in [2n]-[n]$ be a fixed destination column. In [11, 14] the following statement is claimed (section 4.4, time complexities).

For all $i \in [n]$ and for any interval I of $\frac{2n}{k_{i,j}}$ steps the number of packets in rows $0,\ldots,i-1$ that reach their destination column j in interval I is bounded by $\frac{2\sum_{l=0}^{i-1} k_{l,j}}{k_{i,j}} \leq \frac{2n}{k_{i,j}} - 2$.

We now construct a counterexample for the above claim. We assume that n is an even integer. We can place n packets with destination column j in S. We place one packet on each node in the set $\{(x, n-1) \mid x \in [n/2]\} \cup \{(n/2, y) \mid y \in \{0,2,4,\ldots,n-2\}\}$. So we get $k_{i,j} = 1$ for $i \in [n/2]$ and $k_{n/2,j} = n/2$. Obviously $(\star)$ is fulfilled.

The $n/2$ packets in rows $0,\ldots,n/2-1$ reach column j in the same time step. Due to the claim, the number of packets in rows $0,\ldots,n/2-1$ reaching column j in an interval of four time steps is bounded by two. Hence the bound for the number of packets is not correct. Even the order of magnitude is not correct. The reason is that no ceilings are used in the calculation of the bound. A corrected bound for the number of packets is

$$2\sum_{l=0}^{i-1} \left\lceil \frac{k_{l,j}}{k_{i,j}} \right\rceil.$$

We will give no proof of this new bound here. For our example the new bound provides n as an upper bound for the number of packets.

In [12] the basic proof idea used in section 4.2 is similar to that used in [11, 14] but it is a little bit more complicated. A similar statement to that discussed above is obtained. For this statement a counterexample can be constructed analogously.

4.2.2 The Algorithm Presented in [32].

In [32] an oblivious permutation routing algorithm for the two-dimensional mesh of side length n with running time $O(n)$ and buffer size $O(1)$ was presented. The algorithm consists of several parts. One part is called *vertical routing* (section 5 in [32]). In vertical routing a dynamic routing problem on a one-dimensional mesh of size n is solved. In [32] an algorithm for vertical routing was proposed and it was shown that it has a buffer size of at most three. It was claimed that the algorithm solves the problem in $O(n)$ steps but no proof and no proof idea was given. We restrict our attention to the vertical routing and focus our attention to the running time of the algorithm. Details concerning the buffer size are left out. In the following, we describe a simplified version of the problem, give the algorithm, and construct examples such that the algorithm needs $\omega(n)$ steps.

For more details of the algorithm we refer the reader to the original work. The notation we use to describe the problem differs from the notation used in [32].

Description of the Problem.

Given is a one-dimensional mesh of size n and at most n packets. Each node is destination of at most one packet. Let k_i be the number of packets with destination node $> i$ that are inserted into node i, i.e., k_i packets use the edge from i to $i+1$. Every node $i \in [n]$ knows k_i and $k'_i \stackrel{\text{def}}{=} \frac{\sum_{j=0}^{i} k_j}{n}$ (in [32]: $k'_i = \alpha'(\{i, i+1\})$ and $k_i = \alpha(r_i)$, where r_i is the edge on which the packets are injected into node i).

Description of the Algorithm.

Every node $i \in [n]$ maintains a rational state variable $0 \leq q_i < 1$. The initial state of the variable is $q_i = 0$ for all $i \in [n]$. A node sends a packet to its

neighbouring node only if it is *enabled*. A node i is enabled if and only if $q_i + k'_i \geq 1$. In a step a node $i \in [n-1]$ does the following.

- If it is enabled and has some packets, then one of them is sent to $i+1$.
- All incoming packets are stored.
- $q_i = (q_i + k'_i) - \lfloor (q_i + k'_i) \rfloor$. The new value of q_i is used in the next step.

Examples that Needs $\omega(n)$ Steps.

We assume that n is an even integer. The construction uses the fact that the change of the state variable q_i depends on k'_i. Is k'_i a small value, then q_i does not change very much and hence the node i is not enabled for many steps.

Hence we assume that $k_0 = 1, k_2 = 0, \ldots, k_{n-1} = 0$ and that the packet inserted into node 0 has destination $n-1$. For all $i \in [n-1]$, we get $k'_i = 1/n$. The algorithm starts with $t = 0$. Hence at the beginning of step t:

$$q_i(t) = k'_i \cdot t - \lfloor k'_i \cdot t \rfloor.$$

Therefore, each node is enabled in step $c(n-1)$, $c \in \mathbb{N}$ and is not enabled in all other steps. So the packet, inserted into node 0 in step $t = 0$, has to wait $\Omega(n)$ steps at each node and needs $\Omega(n^2)$ steps to reach the destination node.

This problem can be solved by enabling all nodes i for which $k_i = 0$ holds in all steps.

Now assume that $k_0 = 1, \ldots, k_{\frac{n}{2}-1} = 1, k_{\frac{n}{2}} = 0, \ldots, k_{n-1} = 0$ and all packets have a destination in $\{\frac{n}{2}, \ldots, n-1\}$. We get $k'_i = \frac{i+1}{n}$ for $i \in [\frac{n}{2}]$ and $k'_i = \frac{n}{2}$

for $i \in \{\frac{n}{2}, \ldots, n-1\}$. That means a node $i \in [\frac{n}{2}]$ is enabled in at most $i+1$ of n steps and hence a packet has to wait up to $n/(i+1)$ steps in node i, i.e., in total it has to wait $\omega(n)$ steps. For example, choose $n = \prod_{i=1}^{m} i$, $m > 2$. Hence for all $i \in \{1, \ldots, m\}$, we have $\frac{n}{i} \in \mathbb{N}$. Let p be a packet on node 0 in step 0 and assume that p use a link in a node if the node is enabled. It is easy to see that the packet reach node i, $0 < i \leq m$, in step $n \sum_{j=1}^{i} \frac{1}{j}$ (For $i \in \{0, \ldots, m-1\}$, $k_i' | n$ and hence q_i reaches n exactly.). Hence the packet needs $\Omega(n \log m)$ steps to reach node m.

4.3 High Level Structure of the Algorithm.

We divide the network $\mathcal{N}$ into blocks. We have two different kinds of blocks, the *source blocks* and the *destination blocks*. The node sets of the source and destination blocks are both partitions of $V_{\mathcal{N}}$. Every source block has one *exit node* and every destination block has an *entry node* for every source block. A node in a destination block is an entry node for at most one source block. We assume that for every source block directed paths from the exit node to its entry nodes exist such that the paths of a source block form a directed tree. Furthermore, we assume that each edge of $\vec{E_{\mathcal{N}}}$ is used by at most one tree. We call the directed path *connection paths*.

In the algorithm, we arrange the packets within the source blocks such that two packets with the same destination block[6] leave their source block at the exit node with a sufficient large time difference. This is done by the algorithm presented in Section 3.2.2 (gap routing). After a packet has left its source block, it travels to its destination block using a connection path. The packet enters its destination block at an entry node and travels to its

[6] That means that the destination nodes of the two packets belong to the same destination block.

destination node. To solve the problem of edge congestion at the entry nodes we use dynamic routing of Section 3.2.3. The path of a packet within the source and destination block is not simple and hence it is neither a shortest nor an elementary path.

4.4 Oblivious k-k Routing.

4.4.1 Oblivious k-k Routing on Networks.

The aim of this section is to present an oblivious routing algorithm that solves k-k routing problems $(\mathcal{P}, src, dst)$ on a network $\mathcal{N}$. A lower bound for the running time of such an algorithm is $\Omega(k|V_{\mathcal{N}}|^{\frac{1}{2}})$. We do not focus our interest to multiplicative or additive constants. We give an algorithm with running time $O(k|V_{\mathcal{N}}|^{\frac{1}{2}})$ and buffer size $O(k)$.

In this section we need a partitioning of the network $\mathcal{N}$ into blocks.

Definition 4.2 (Partitioning of $\mathcal{N}$) *Let $\mathcal{N} = (V, E)$ be a network, $t_1, t_2 \in \mathbb{N}$. We call $((S_i)_{i\in[t_1]}, (D_i)_{i\in[t_2]}, (entry_i)_{i\in[t_2]}, (V_i, E_i, exit_i)_{i\in[t_1]})$ a (t_1, t_2)-partitioning of $\mathcal{N}$ if and only if*

1. *$(S_i)_{i\in[t_1]}$ and $(D_i)_{i\in[t_2]}$ are two partitions of V such that*
   - *$\forall i \in [t_1]$: The graph $\mathcal{N}_{S,i} \stackrel{\text{def}}{=} (S_i, E \cap \mathcal{P}_2(S_i))$ is connected.*
   - *$\forall i \in [t_2]$: The graph $\mathcal{N}_{D,i} \stackrel{\text{def}}{=} (D_i, E \cap \mathcal{P}_2(D_i))$ is connected.*
2. *$\forall i \in [t_2]$: $entry_i : [t_1] \longrightarrow D_i$ is an injective mapping.*
3. *$\forall i \in [t_1]$: $T_i \stackrel{\text{def}}{=} (V_i, E_i, exit_i)$ is a directed tree with node set V_i, edge set E_i, and root $exit_i$ such that*
   - *$exit_i \in S_i$,*

- $V_i \subseteq V$, $E_i \subseteq \vec{E}$, *and*
- $\forall j \in [t_2] : entry_j(i) \in V_i$.

4. $\forall e \in \vec{E} : |\{i \in [t_1] \mid e \in E_i\}| \leq 1$

We call the subnetworks $\mathcal{N}_{S,i}$ of $\mathcal{N}$ source blocks and the subnetworks $\mathcal{N}_{D,i}$ of $\mathcal{N}$ destination blocks. In the following we write S_i instead of $\mathcal{N}_{S,i}$ and D_i instead of $\mathcal{N}_{D,i}$.

For $i \in [t_1], j \in [t_2]$, we call the directed path in T_i from root $exit_i$ to $entry_j(i)$ the connection path from S_i to D_j, $entry_j(i)$ entry node of S_i in D_j, $exit_i$ exit node of S_i, the nodes in $entry_j([t_1])$ entry nodes, and the nodes $exit_0, \ldots, exit_{t_1-1}$ exit nodes.

Let $s_1 = \max\{|S_i| \mid i \in [t_1]\}$, $s_2 = \max\{|D_i| \mid i \in [t_2]\}$, and $s_3 = \max\{\max\{d_{T_i}(exit_i, x) \mid x \in V_i\} \mid i \in [t_1]\}$. We call (s_1, s_2, s_3) the size of the partitioning.

In the rest of this section we assume that a (t_1, t_2)-partitioning of size (s_1, s_2, s_3) of $\mathcal{N}$ exists and that $\mathcal{N}$ is of fixed degree, i.e. $deg(\mathcal{N}) \in O(1)$. A shortest (directed) path from node x to node y in $\mathcal{N}$ is denoted by $x \Longrightarrow^* y$. Each source and destination block of $\mathcal{N}$ is connected. Let n be the size of a block. We are able to embed a one-dimensional mesh of the size n with load 1, congestion 2, and dilation 3 into the block (see Section 3.2.1). Using a step by step simulation, any algorithm designed for a one-dimensional mesh of size n can be performed on a block of the same size with a slowdown of $sd \in O(1)$ and additional buffer size $ab \in O(1)$ (Theorem 3.3). In a step by step simulation, using an embedding Φ, all packets sent from node i to node $i+1$ or sent from node $i+1$ to node i, $i \in [n-1]$, use path $\Phi(\{i, i+1\})$ in the block. Hence an algorithm remains oblivious if we use such a simulation. So we can assume that S_i, $i \in [t_1]$, (D_i, $i \in [t_2]$) is a one-dimensional mesh of size $|S_i|$ ($|D_i|$).

Let $X \in \{S, D\}$ and let v, v' be two nodes of block X_i, $|X_i| = n$. In the following algorithm, we route a packet p on a cycle from node v to v in X_i. This means, that p is transported along the cycle

$$v \Longrightarrow^* n-1 \Longrightarrow^* 0 \Longrightarrow^* v$$

in X_i.

We also route p from v to v' in X_i. If $v \leq v' \leq n-1$, this means that p is transported along the path

$$v \Longrightarrow^* v'.$$

If $0 \leq v' < v$, p is transported along the path

$$v \Longrightarrow^* n-1 \Longrightarrow^* v'.$$

The definition of the partitioning of $\mathcal{N}$ allows that an edge is used by a source block, a destination block, and a connection path. To avoid edge congestion, packets are transported in only one of these three structures in a step in part two and three of our routing algorithm. Packets in source blocks are transported in steps $\equiv 0 \pmod 4$, packets in connection paths are routed in steps $\equiv 1 \pmod 4$, and packets in destination blocks are moved in steps $\equiv 2 \pmod 4$ and $\equiv 3 \pmod 4$.

The oblivious k-k routing algorithm consists of three parts. In part 1, packets are transported within the source blocks. In all source blocks S_i and for all destination blocks D_j, the number of packets in S_i with a destination in D_j is calculated. This value is used in the second and third part. In part 2, every source block sends at most one packet to every destination block. The values calculated in part 1 are transported to the destination blocks by these packets. In part 3, an instance of gap routing in the source blocks and an instance of dynamic routing in the destination blocks is solved. The

gap routing ensures that any two packets reach their destination block with a sufficient large time difference and so dynamic routing can work properly. Furthermore, in part 3, two kinds of packets move in the destination blocks. We call them *phase 1* and *phase 2* packets. These two kinds of packets compete for the edges in the destination blocks. To avoid congestion, phase 1 packets move in steps $\equiv 2 \pmod 4$ and phase 2 packets move in steps $\equiv 3 \pmod 4$. The packets routed by dynamic routing in part 3 are phase 1 packets. Therefore, in part 3, the dynamic routing algorithm is active in steps $\equiv 2 \pmod 4$ and the gap routing algorithm is active in steps $\equiv 0 \pmod 4$.

The k-k Routing Algorithm for Network $\mathcal{N}$. We denote the number of packets in source block S_i with destination block D_j by $m_{i,j}$ and set $n_{i,j} \stackrel{\text{def}}{=} \max\{0, m_{i,j} - 1\}$. If $src(p) \in S_i$, then $sb(p) = i$ and if $dst(p) \in D_j$, then $db(p) = j$.

Algorithm KKOblivious.

1. In every source block S_i, send every packet p on a cycle from $src(p)$ to $src(p)$. During this routing, every node $s \in S_i$ determine $n_{sb(p),db(p)}$ by counting, for $p \in src^{-1}(\{s\})$. Store $n_{sb(p),db(p)}$ in the additional information field of p. Furthermore, in every source block S_i and for every destination block D_j, choose a packet $p_{i,j}$ such that $db(p_{i,j}) = j$ (if one exists).

2. In every source block S_i and for every destination block D_j, send packet $p_{i,j}$ (if it exists) to node $dst(p_{i,j})$ along the following path: Route packet $p_{i,j}$ from node $src(p_{i,j})$ to node $exit_i$ in S_i. If $p_{i,j}$ reaches $exit_i$, route it to node $entry_j(i) \in D_j$ in such a way that only edges of the connection path from S_i to D_j are used. In destination block D_j route packet $p_{i,j}$ on a cycle from $entry_j(i)$ to $entry_j(i)$. In the

beginning of the cycle $p_{i,j}$ is a phase 1 packet until it reaches node 0 (In the case $entry_j(i) = 0$ the packet $p_{i,j}$ is a phase 1 packet until it reaches node 0 for the second time). Afterwards $p_{i,j}$ is a phase 2 packet. During the routing on a cycle $p_{i,j}$ delivers $n_{i,j}$ to nodes in the set $entry_j([t_1])$ and the nodes in D_j calculate values for dynamic routing (this is described later). Route $p_{i,j}$ from $entry_j(i)$ to $dst(p_{i,j})$ in D_j.

3. In every source block S_i solve the following instance of gap routing. Here $d_i : \mathbb{N}_0 \longrightarrow \mathbb{N}$ is the function that gives the lower bound for the time difference in block S_i. There are t_2 classes. A packet p belongs to class $j \in [t_2]$ if and only if $dst(p) \in D_j$.

Instance of gap routing

- The destination of a packet is to reach the node $|S_i| - 1$ and to leave it.
- The source of a packet p is node $src(p)$. Hence there are at most k packets on each node initially.
- For $j \in [t_2]$, we set $d_i(j) = \left\lfloor \frac{4ks_2}{\langle n_{i,j} \rangle} \right\rfloor$, if $n_{i,j} > 0$.

If a packet in S_i with a destination in D_j leave node $|S_i| - 1$, then route it (without delay) to node $exit_i$ in S_i (If $|S_i| - 1 = exit_i$, then route it to node $entry_j(i)$ using the edges of the connection path from S_i to D_j). If it reaches $exit_i$, route it to its destination along the same path as the packet in part 2, i.e., route it to node $entry_j(i)$ in D_j using the edges of the connection path from S_i to D_j. In D_j route it on a cycle from $entry_j(i)$ to $entry_j(i)$ (the routing from node $entry_j(i)$ to node $|D_j| - 1$ is done by dynamic routing) and from $entry_j(i)$ to its destination node. As in part 2 the packets are phase 1 packets until

they reach node 0 in D_j (in the case $entry_j(i) = 0$ until they reach node 0 for the second time). Afterwards they are phase 2 packets.

Route packets, that reach node $entry_j(i)$ in D_j, to node $|D_j| - 1$ by solving the following instance of dynamic routing.

Instance of Dynamic Routing

- The nodes in D_j simulate the injectors. They simulate a set of (k'_j, T'_j, T_j) bounded injectors, where
  - $k'_j = \sum_{l \in [t_1]} n_{l,j}$,
  - $T'_j = 4\langle k'_j \rangle$, and
  - $T_j = O(ks_1 + ks_2 + s_3)$.

  A node $l \in D_j$ such that $l = entry_j(i)$ for $i \in [t_1]$ simulate an injector which injects $n_{i,j}$ packets into node l.
- The destination of a packet is node $|D_j| - 1$ in D_j.

(Remark: A packet in destination block D_j that reaches node $|D_j| - 1$ in dynamic routing is routed to node 0 without delay.)

Technical Details and Analysis of the Algorithm.

We assume that all nodes know s_1, s_2, s_3, and k and that they are able to determine the source and destination block of a packet from its source and destination address field. The additional information field of a packet is used in different ways.

- Information whether a packet is routed within a soure block, destination block, or along a connection path is stored.
- Information whether a packet is a phase 1 or a phase 2 packet is stored.

- The value of $n_{i,j}$ is stored.

The additional information field of a packet is used by the nodes for routing decisions. The routing decisions for packets moving within the blocks are simple. In a block of size n a node does the following with packets moving within the block: node 0 sends all packets to node 1, node $0 < i < n-1$ sends all packets coming from node $i-1$ to node $i+1$ and all packets coming from node $i+1$ to node $i-1$, and node $n-1$ sends all packets to node $n-2$. The generalization to the case that an embedding Φ is used is clear and hence omitted. The routing decisions for packets moving along a connection path or for packets that have to leave their (source) block can be done by the nodes using a table with an entry for each possible destination block. Packets that reach their entry node, say node i, in a (destination) block are moved (after their injection) in direction of node $n-1$ (0 if $i = n-1$).

Part 1:

In the beginning at most k packets are on a node. In every source block S_i, $i \in [t_1]$, every packet is sent on a cycle from its source to its source. If in S_i a packet with destination block D_j, $j \in [t_2]$ exists, then one of these packets can be chosen by a node, e.g., node 0 choose the first packet destined for a destination block, which visits node 0, by setting a bit in the additional information field of the packet. To do this, node 0 has to store at most one bit per class. The number of bits can be reduced to one, provided there at least t_2 nodes in the source block. In this case a node chooses at most one packet.

Part 1 can be done in at most $O(ks_1)$ steps and with a buffer size of $k + ad$ ($+ad$ due to the simulation).

Part 2:

In a source block S_i, $i \in [t_1]$, the packets $p_{i,j}$ travel to their destination by

using the paths described in the algorithm. The length of such a path is at most $2s_1 + s_3 + 4s_2$. During this routing congestion occurs at the entry places. Here we give priority to packets traveling within the destination block. At most ks_2 packets enter a destination block in part 2. Hence a packet is blocked at most ks_2 times. Therefore part 2 can be done in at most $O(s_1 + s_2 + s_3 + ks_2)$ steps.

A node belongs to a source and a destination block. Furthermore, it can be node in at most $deg(\mathcal{N})$ trees and it can be an entry node. At most k packets are stored in a node initially, ad additional packets can be on a node due to the simulation, two additional packets can be on a node in a source block, $deg(\mathcal{N})$ additional packets can be on a node due to the trees, four additional packets can be on a node in a destination block (two phase 1 packets and two phase 2 packets), and one additional packet can be on an entry node. Therefore, at most $k + ad + deg(\mathcal{N}) + 8$ packets are on a node in a step.

In part 2, values for dynamic routing in part 3 are calculated. We assume that the destination block D_j has n nodes. In a destination block the entry nodes have two jobs. They are nodes in a block and they insert packets into a block. An entry node l inserts packets into node l. In a destination block, a node that is not an entry node is an injector that inserts no packets. Such injectors can be ignored. We call an injector that inserts no packet into a block *inactive* and an injector that inserts packets into a block *active*. Also entry nodes can be inactive. An active injector I_l, $l \in [n]$ needs to know η_l, δ_l and $\gamma_{\bar{l}}$ to work properly. (Remember, $\gamma_{\bar{l}}$ is the interval length, $\delta_l \cdot \gamma_{\bar{l}}$ is the creation time of I_l, and η_l describes the intervals in which I_l is enabled.) This values can be calculated by the node using the information carried by packets $p_{i,j}$. We remind $p_{i,j}$ is routed on a cycle from $entry_j(i)$ to $entry_j(i)$ in D_j and transports $n_{i,j}$. Let I_l be an active injector in D_j

such that $entry_j(i) = l$. We have

- $\gamma_{\bar{l}} = \max\{\frac{T'}{\langle n_{i,j}\rangle} \mid i \in [t_1]\}$, where $T' = 4\langle\sum_{l\in[t_1]} n_{l,j}\rangle$, and
- $\delta_l = \frac{\max\{\langle n_{i,j}\rangle | i\in[t_1]\}}{\langle n_{i,j}\rangle}$.

It is easy to see node l in D_j can compute $\gamma_{\bar{l}}$ and δ_l using $n_{i,j}, i \in [t_1]$.

The value of η_l is computed using the method described on pages 35 to 38. In the following, we use the notation of Definition 3.24 and describe how η_l is computed. Let I_l be an active injector in D_j such that $entry_j(i) = l$ and $\langle n_{i,j}\rangle = 2^{j'}$ for a $j' \in \mathbb{N}_0$. I_l is an injector of $U_{j'}$ in D_j. For all injectors t of $U_{j'}$ we have $\delta_t = \theta_{j'}$. Node l counts $|U_{j'}|$ and sets $\theta_{j'} = \delta_l$. Furthermore, it counts $|\{l' \in U_{j'} \mid l' < l\}|$ to get an unique number in $[|U_{j'}|]$. We denote this number by u_l. With the help of $\theta_{j'}$, $|U_{j'}|$ and u_l, each node l decides whether it is a normal injector or a special injector of $U_{j'}$. If $u_l < \theta_{j'} \left\lfloor \frac{|U_{j'}|}{\theta_{j'}} \right\rfloor$, then l is a normal injector with number u_l. Otherwise, it is a special injector with number $u_l \bmod \delta_l$. In the case l is a normal injector, it sets $\eta_l = u_l \bmod \delta_l$. In the case l is a special injector, node l computes η_l as described on pages 36 to 38. To do this, node l has to compute $b_{\bar{j}}, \ldots, b_{j'}$. To compute $b_{\bar{j}}, \ldots, b_{j'}$ it computes $r_{\bar{j}}, \ldots, r_{j'}$. To compute $r_{\bar{j}}, \ldots, r_{j'}$ it counts $|U_{\bar{j}}|, \ldots, |U_{j'}|$.

Part 3:

We begin with an analysis of gap routing in S_i, $i \in [t_1]$. Due to Theorem 3.10 this instance of gap routing can be done on a one-dimensional mesh of size $|S_i|$ in at most $|S_i| + k|S_i| + 8ks_2 \leq 10k \max\{s_1, s_2\}$ steps using a buffer size of at most $k + 1$. Hence $O(k(s_1 + s_2))$ steps and a buffer size of $k + ad + 1$ is needed.

Now we analyse dynamic routing in D_j, $j \in [t_2]$. We have to check whether the instance of dynamic routing in D_j is well defined. We check the definitions 3.13, 3.14 and whether the nodes are able to simulate a set of (k'_j, T'_j, T_j) bounded injectors, where $k'_j = \sum_{l\in[t_1]} n_{l,j}$, $T_j = O(k(s_1 + s_2) + s_3)$, and

$T'_j = 4\langle k'_j \rangle$. Obviously k'_j packets (in part 3) have a destination in block D_j. Let l be an entry node of D_j such that $entry_j(i) = l$ and $n_{i,j} > 0$. We have to check wether l is able to simulate a restricted $(n_{i,j}, O(k(s_1+s_2)+s_3), \frac{4\langle k'_j \rangle}{\langle n_{i,j} \rangle})$ injector with offset $x \in |D_i|$. Exactly $n_{i,j}$ packets enter block D_j at l. Hence l is able to create $n_{i,j}$ packets. We have $n_{i,j} \leq k'_j$ and so $\frac{4\langle k'_j \rangle}{\langle n_{i,j} \rangle} \in \mathbb{N}$. Gap routing needs $O(k(s_1+s_2))$ steps and a connection path has a length of at most s_3. Therefore, the last packet reaches l after at most $O(k(s_1+s_2)+s_3)$ steps and so the last packet can be created by l after at most $O(k(s_1+s_2)+s_3)$ steps, provided at most $O(k(s_1+s_2)+s_3)$ steps elapse between the arrival of a packet and its creation. It is clear that node l is able to simulate an injector with any creation rate and any offset, provided it is able to store an unbounded number of packets.

So the instance is well defined and the nodes of D_j are able to simulate the injectors, provided the node can store an unbounded number of packets.

Now we consider how many packets node l have to store to simulate an injector. If two packets with source in S_i and destination in D_j reach node l in steps t_1 and t_2, then $|t_2 - t_1| \geq \left\lfloor \frac{4ks_2}{\langle n_{i,j} \rangle} \right\rfloor$ due to gap routing in S_i. We have $k'_j \leq ks_2$. Therefore, in any period of $\frac{4\langle k'_j \rangle}{\langle n_{i,j} \rangle}$ consecutive steps, at most one packet with source in S_i and destination in D_j reaches node l. Hence node l need at most a buffer size of two to simulate a restricted injector with offset l and creation rate $\frac{4\langle k'_j \rangle}{\langle n_{i,j} \rangle}$. It has to store one packet that waits to be created and one packet that waits to be inserted.

We have $T'_j \geq 2k'_j$. Due to Corollary 3.32 at most $2T_j + |D_j|$ steps are needed to solve the instance of dynamic routing on a one-dimensional mesh of size $|D_j|$. We have $2T_j + |D_j| = O(k(s_1+s_2)+s_3)$.

After dynamic routing, a packet has to travel along a path of at most $O(s_3)$ length. Hence all packets have reached their destination after $O(k(s_1+s_2)+s_3)$ steps.

The maximal number of packets on a node in a step in part 3 is $k + ad + deg(\mathcal{N}) + 9$ (see part 2: +1 at the entry nodes).

This yields

Theorem 4.3 *Let $\mathcal{N}$ be a network of fixed degree for which a (t_1, t_2)-partitioning of size (s_1, s_2, s_3) exists.*

1. *Algorithm* **KKOblivous** *is a k-k routing algorithm on $\mathcal{N}$. It has a running time of*

$$O(k(s_1 + s_2) + s_3) \text{ steps}$$

   and a buffer size of $O(k)$.

2. *Algorithm* **KKOblivious** *is an oblivious k-k routing algorithm.*

Proof:

1. Follows from the above discussion.

2. In part 1, each packet cycles from its source node to its source node in its source block. In part 3 (or part 2), each packet moves from its source node to the exit node of its source block, then it moves to an entry node in its destination block using a connection path, in the destination block it cycles one time from the entry node to the entry node and then it moves to its destination. Hence the path of a packet only depends on its source node and its destination node.

◇

Now we discuss some further aspects of the routing algorithm. As mentioned above the nodes calculate several values. The calculation of η for special injectors in part 2 of the algorithm needs $O(ks_2)$ bits. In part 1, the choice

of $p_{i,j}$ need $O(t_2)$ bits and the calculation of $n_{i,j}$ need $O(k \min\{s_1, s_2\})$ bits. In part 3, at most $O(\log ks_2)$ bits are needed per node (see Theorem 3.10 and Theorem 3.31). Hence at most $O(ks_2 + t_2 + \log ks_1)$ bits are needed per node.

Finally, we analyse the size of the additional information field. As seen in the above discussion, we store $n_{i,j}$ in this field. Hence $O(\log k \min\{s_1, s_2\})$ bits are needed. We can reduce its size to $O(\log \min\{s_1, s_2\})$, if we send $n_{i,j} \geq \min\{s_1, s_2\}$ by using up to k packets. Each of these packets transports a value of at most $\lceil \frac{n_{i,j}}{k} \rceil$. The (worst case) running time is not affected, but we have to store up to k packets in an entry node in part 2. Hence a buffer size of $2k + ad + 7$ is needed in part 2.

This yields

Theorem 4.4 (Oblivious Routing on $\mathcal{N}$.) *Let $\mathcal{N}$ be a network of fixed degree for which a (t_1, t_2)-partitioning of size (s_1, s_2, s_3) exists. Then an oblivious k-k routing algorithm on $\mathcal{N}$ with running time $O(k(s_1 + s_2) + s_3)$ and buffer size $O(k)$ exists. The algorithm needs at most $O(ks_2 + t_2 + \log ks_1)$ bits on each node for calculations and at most $O(\log |V_{\mathcal{N}}|)$ bits in the additional information field of each packets to store information.*

4.4.2 Oblivious k-k Routing on r-Dimensional Meshes.

If a partitioning of $M_{r,n}$ of size $(O(n^{\frac{r}{2}}), O(n^{\frac{r}{2}}), O(n^{\frac{r}{2}}))$ exists, then an $O(kn^{\frac{r}{2}})$ oblivious k-k routing algorithm for $M_{r,n}$ using $O(k)$ buffer size exists. We specify such a partitioning of $M_{r,n}$ in this section. We distinguish two cases, even $r > 1$ and odd $r > 1$. For $r = 1$ oblivious k-k routing can be done in $O(kn)$ steps with a buffer size of $O(k)$ by an algorithm using shortest paths and the farthest destination first queueing strategy [47].

In this section $x \Longrightarrow^* y$ denotes a shortest directed path from node x to

node y in $\vec{M}_{r,n}$.

Even $r > 1$.

We give an $(n^{\frac{r}{2}}, n^{\frac{r}{2}})$-partitioning of $M_{r,n}$. We use the set $[n]^{\frac{r}{2}}$ as index set for the source blocks, destination blocks, entry places, and exit places.

We begin with the case $r = 2$.

1. For $i \in [n]$, we set $S_i \stackrel{\text{def}}{=} \{(i, x) \mid x \in [n]\}$.
2. For $i \in [n]$, we set $D_i \stackrel{\text{def}}{=} \{(x, i) \mid x \in [n]\}$.
3. For $i, j \in [n]$, we set $entry_j(i) \stackrel{\text{def}}{=} (i, j) \in D_j$.
4. For $i \in [n]$, we set $T_i \stackrel{\text{def}}{=} (S_i, E_i, exit_i)$, where $exit_i = (i, n-1)$ and $E_i = \{((i, x), (i, x-1)) \mid x \in \{1, \ldots, n-1\}\}$.

It is easy to see that $((S_i)_{i\in[n]}, (D_i)_{i\in[n]}, (entry_i)_{i\in[n]}, (T_i)_{i\in[n]})$ is an (n, n)-partitioning of $M_{2,n}$ of size $(n, n, n-1)$.

Example:

Figure 4.1 shows a $(4, 4)$-partitioning of $M_{2,4}$ of size $(4, 4, 3)$. In the upper right corner the figure shows the four source blocks $S_0, \ldots, S_3$ and the four exit places $exit_0, \ldots, exit_3$ (black boxes). Each source block is built by a row of the mesh and consists of four nodes. In the lower right corner the figure shows the four destination blocks $D_0, \ldots, D_3$ and the entry nodes $entry_0(3), \ldots, entry_3(3)$ (black circles). Each destination block is built by a column of the mesh and consists of four nodes. In the lower left corner the figure shows the four trees $T_0, \ldots, T_3$. Four connection paths are shown. In tree T_i the connection path from $exit_i$ to $entry_0(i)$ is shown. All other connection paths are subpaths of these four paths.

In Figure 4.2 the path of a packet with source node s and destination node d is shown. On the left side the path of a packet in part 1 of the routing

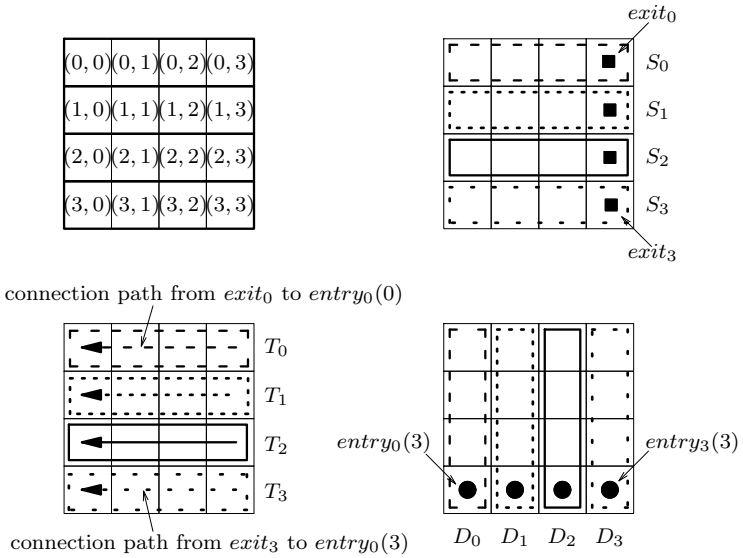

Figure 4.1: A $(4,4)$-partition of size $(4,4,3)$ of $M_{2,4}$.

algorithm is shown. It is possible that a packet travels to its destination in part 2 or in part 3 of the algorithm. In both cases the paths are equal. The path is shown on the right side.

$\diamond$

In the literature, the permutation routing problem on two-dimensional meshes is one of the most studied problems. We calculate the running time of **KKOblivious** for this case. We have $r = 2$ and $k = 1$. The paths in the source and destination blocks are edge disjoint. The connection paths and the paths in the destination blocks are edge disjoint, too. So we are able to move the packets in the source blocks within steps $\equiv 0 \,(\mathrm{mod}\, 2)$, along the connection paths in steps $\equiv 1 \,(\mathrm{mod}\, 2)$ and in the destination blocks in steps $\equiv 0 \,(\mathrm{mod}\, 2)$ and $\equiv 1 \,(\mathrm{mod}\, 2)$. We get the following running times.

1. Part 1 can be done in $2n - 2$ steps.

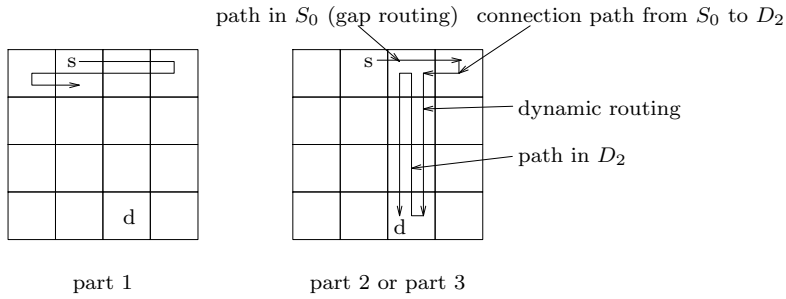

Figure 4.2: Path of a packet in oblivious routing.

2. The length of a path of a packet in part 2 is at most $6n$. A packet can be blocked by at most n packets at an entry node. Hence part 2 can be done in at most $14n$ steps.

3. In source block S_i we have $d_i(j)n \leq 8n$, for all $j \in [n]$. Hence gap routing in part 3 needs at most $2(1+1+8)n$ steps (Theorem 3.10). The connection path has length n. So the last packet reaches an entry node at most $22n$ steps after the beginning of part 3. The injectors have a creation time of at most $4n$ and hence the last packet can be inserted at most $22n+2(4n) = 30n$ steps after the beginning of part 3. After the insertion a packet has to travel along a path of at most $4n$ length to get to its destination. Note that we can move the packet in each step. Hence the last packet reaches its destination after at most $34n$ steps.

Algorithm **KKOblivious** needs at most $50n$ steps and a buffer size of 10 on $M_{2,n}$, if $k = 1$. The fastest permutation routing algorithm on $M_{2,n}$ needs $2n - 2$ steps and has a buffer size of 32 (+4 for passing packets) [46]. The number of steps of this algorithm is optimal. The oblivious permutation routing algorithm presented in [12] achieve a running time of approximately $3n$ steps with a buffer size of > 100000 and a running time of $20n$ steps with

a buffer size[7] of 304.

Now we consider the case $r > 2$.

1. For $i \in [n]^{\frac{r}{2}}$, we set $S_i \stackrel{\text{def}}{=} \{(i, x) \in [n]^r \mid x \in [n]^{\frac{r}{2}}\}$.
2. For $i \in [n]^{\frac{r}{2}}$, we set $D_i \stackrel{\text{def}}{=} \{(x, i) \in [n]^r \mid x \in [n]^{\frac{r}{2}}\}$.
3. For $i, j \in [n]^{\frac{r}{2}}$, we set $entry_j(i) \stackrel{\text{def}}{=} (i, j) \in D_i$.
4. For $i \in [n]^{\frac{r}{2}}$, we set $T_i \stackrel{\text{def}}{=} (S_i, E_i, exit_i)$ and $exit_i = (i, n-1, \ldots, n-1) \in [n]^r$. We define T_i as union of directed paths from $exit_i$ to $entry_j(i)$[8], $j \in [n]^{\frac{r}{2}}$. Let $j = (j_0, \ldots, j_{\frac{r}{2}-1})$. The path from $exit_i$ to $entry_j(i)$ is given as follows.

$$
\begin{aligned}
exit_i &= \\
(i, \underbrace{n-1, \ldots, n-1}_{r/2}) &\Longrightarrow^* \\
(i, j_0, \underbrace{n-1, \ldots, n-1}_{r/2-1}) &\Longrightarrow^* \\
(i, j_0, j_1, n-1, \ldots, n-1) &\Longrightarrow^* \\
\vdots &\Longrightarrow^* \\
(i, j_0, \ldots, j_{\frac{r}{2}-2}, n-1) &\Longrightarrow^* \\
(i, j_0, \ldots, j_{\frac{r}{2}-2}, j_{\frac{r}{2}-1}) &= \\
(i, j) = entry_j(i).
\end{aligned}
$$

It is easy to see that $((S_i)_{i\in[n]}, (D_i)_{i\in[n]}, (entry_i)_{i\in[n]}, (T_i)_{i\in[n]})$ is an $(n^{\frac{r}{2}}, n^{\frac{r}{2}})$-partitioning of $M_{r,n}$ of size $(n^{\frac{r}{2}}, n^{\frac{r}{2}}, \frac{r}{2}(n-1))$.

[7]This is the smallest possible buffer size of their algorithm. We use our model to calculate this buffer size. In their model the corresponding value is 38.

[8]The directed path from $exit_i$ to $entry_j(i)$ is the connection path from S_i to D_j

Odd $r > 1$.

We set $r' = \frac{r-1}{2}$, $m = \lceil \sqrt{n} \rceil$ and $s = \lfloor \frac{n}{m} \rfloor$. Obviously $s \leq m$ and $m \in O(n^{\frac{1}{2}})$. We use the set $[n]^{r'} \times [s]$ as index set for the source blocks, destination blocks, entry places, and exit places.

We begin with the case $r = 3$. Hence $r' = 1$.

1. For $i \in [n]$, $j \in [s-1]$, we set

$$S_{(i,j)} \stackrel{\text{def}}{=} \{(i, y, x) \mid x \in [n],\, y \in [n],\, jm \leq y < (j+1)m\}.$$

   It is $|S_{(i,j)}| = mn = O(n^{\frac{3}{2}})$.

2. For $i \in [n]$, we set

$$S_{(i,s-1)} \stackrel{\text{def}}{=} \{(i, y, x) \mid x \in [n],\, y \in [n],\, (s-1)m \leq y < n\}.$$

   It is $|S_{(i,s-1)}| = mn + (n \mod m) \cdot n = O(n^{\frac{3}{2}})$.

3. For $i \in [n]$, $j \in [s-1]$, we set

$$D_{(i,j)} \stackrel{\text{def}}{=} \{(x, y, i) \mid x \in [n],\, y \in [n],\, jm \leq y < (j+1)m\}.$$

   It is $|D_{(i,j)}| = mn$.

4. For $i \in [n]$, we set

$$D_{(i,s-1)} \stackrel{\text{def}}{=} \{(x, y, i) \mid x \in [n],\, y \in [n],\, (s-1)m \leq y < n\}.$$

   It is $|D_{(i,s-1)}| = mn + (n \mod m) \cdot n$.

5. For $i, i' \in [n]$, $j, j' \in [s]$, we set

$$entry_{(i',j')}(i, j) \stackrel{\text{def}}{=} (i, j'm + j, i') \in D_{(i',j')}.$$

6. For $i \in [n]$, $j \in [s]$, we set $T_{(i,j)} = (V_{(i,j)}, E_{(i,j)}, exit_{(i,j)})$. We define $T_{(i,j)}$ as union of directed paths from $exit_{(i,j)}$ to $entry_{(i',j')}(i,j)$, $i' \in [n]$, $j' \in [s]$. The root $exit_{(i,j)}$ of $T_{(i,j)}$ is the node (i, jm, j). For $i, i' \in [n]$, $j, j' \in [s]$ the path from $exit_{(i,j)}$ to $entry_{(i',j')}(i,j)$, is given as follows

$$\begin{aligned} exit_{(i,j)} &= \\ (i, jm, j) &\Longrightarrow^* \\ (i, j'm + j, j) &\Longrightarrow^* \\ (i, j'm + j, i') &= \\ entry_{(i',j')}(i,j). & \end{aligned}$$

It is not hard to see that

$$((S_{(i,j)})_{(i,j)\in[n]\times[s]}, (D_{(i,j)})_{(i,j)\in[n]\times[s]}, (entry_{(i,j)})_{(i,j)\in[n]\times[s]}, (T_{(i,j)})_{(i,j)\in[n]\times[s]})$$

is an (sn, sn)-partitioning of $M_{3,n}$ of size

$$(mn + (n \bmod m) \cdot n, mn + (n \bmod m) \cdot n, (s-1)m + (n-1)).$$

Example:

If $n = 16$, then $m = s = 4$. We give an example of a $(64, 64)$-partitioning of $M_{3,16}$ of size $(64, 64, 27)$.

Figure 4.3 shows the source and destination blocks of the partitioning.

Figure 4.4 shows a subnetwork of $M_{3,16}$ consiting of nodes nodes $\{(i, x, y) \mid x, y \in [16]\}$, $i \in [16]$. It shows the four source blocks $S_{i,0}, \ldots, S_{i,3}$ and the four exit nodes $exit_{(i,0)}, \ldots, exit_{(i,3)}$ in the subnetwork. Figure 4.5 shows connection paths from $exit_{(i,2)}$ to $entry_{(i',j')}(i,2)$, where $i' \in \{0, 15\}$ and

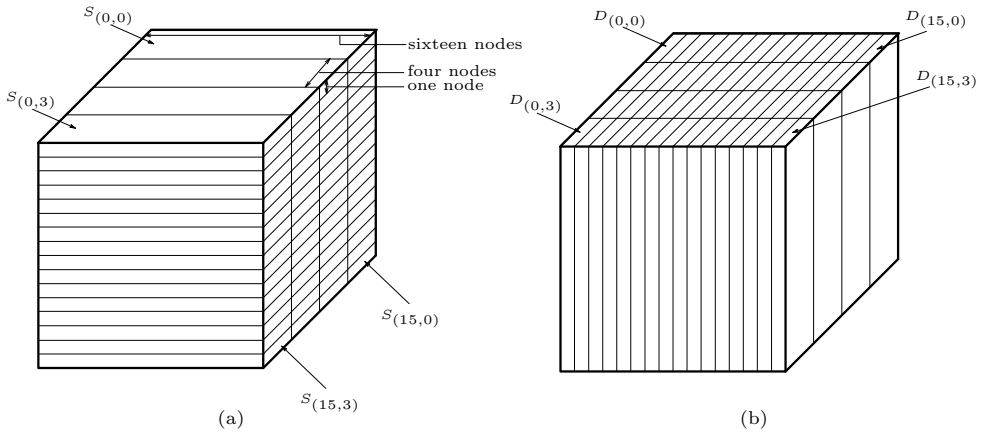

Figure 4.3: Source and destination blocks in $M_{3,15}$.
(a) Source blocks (b) Destination blocks.

$j' \in \{0,1,2,3\}$. All other connection paths from $exit_{(i,2)}$ to $entry_{(i',j')}(i,2)$, $i' \in [15]$, $j' \in [4]$ are subpaths of these paths. Hence these connection paths build tree $T_{(i,2)}$. Figure 4.6 shows all connection paths in the subnetwork. Note that the directed trees are edge disjoint but not node disjoint.

$\diamond$

Now we consider the case $r > 3$. Hence $r' > 1$.

1. For $i \in [n]^{r'}$, $j \in [s-1]$, we set

$$S_{(i,j)} \stackrel{\text{def}}{=} \{(i,y,x) \mid x \in [n]^{r'},\ y \in [n],\ jm \leq y < (j+1)m\}.$$

   It is $|S_{(i,j)}| = mn^{r'} = O(n^{\frac{r}{2}})$.

2. For $i \in [n]^{r'}$, we set

$$S_{(i,s-1)} \stackrel{\text{def}}{=} \{(i,y,x) \mid x \in [n]^{r'},\ y \in [n],\ (s-1)m \leq y < n\}.$$

   It is $|S_{(i,s-1)}| = mn^{r'} + (n \mod m)n^{r'} \in O(n^{\frac{r}{2}})$.

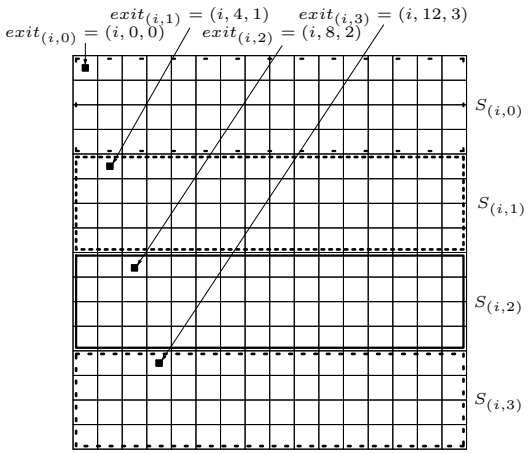

Figure 4.4: Source blocks and exit nodes.

In this figure $i \in [16]$. The figure shows a subnetwork of $M_{3,16}$ consisting of nodes $\{(i, x, y) \mid x, y \in [16]\}$. It shows source blocks $S_{(i,0)}, \ldots, S_{(i,3)}$ and exit nodes $exit_{(i,0)}, \ldots, exit_{(i,3)}$.

3. For $i \in [n]^{r'}$, $j \in [s-1]$, we set

$$D_{(i,j)} \stackrel{\text{def}}{=} \{(x, y, i) \mid x \in [n]^{r'},\ y \in [n],\ jm \leq y < (j+1)m\}.$$

It is $|D_{(i,j)}| = mn^{r'}$.

4. For $i \in [n]^{r'}$, we set

$$D_{(i,s-1)} \stackrel{\text{def}}{=} \{(x, y, i) \mid x \in [n]^{r'},\ y \in [n],\ (s-1)m \leq y < n\}.$$

It is $|D_{(i,s-1)}| = mn^{r'} + (n \mod m)n^{r'}$.

5. For $i, i' \in [n]^{r'}$, $j, j' \in [s]$, we set

$$entry_{(i',j')}(i, j) \stackrel{\text{def}}{=} (i, j'm + j, i') \in D_{(i',j')}.$$

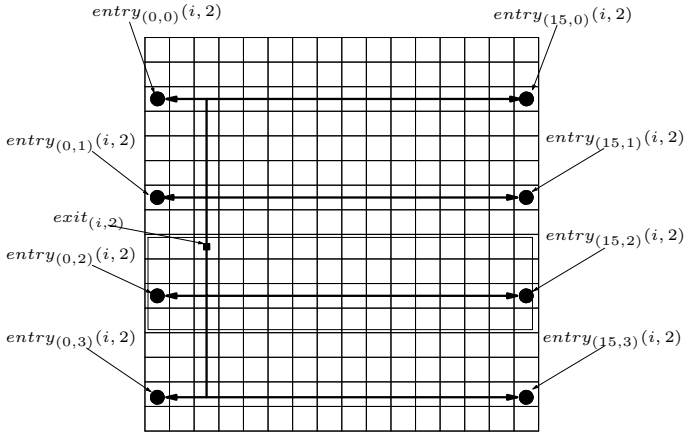

Figure 4.5: Connection paths from $exit_{(i,2)}$ to entry nodes.

This figure shows connection paths from $exit_{(i,2)}$ to $entry_{(i',j')}(i,2)$, where $i' \in \{0,15\}$ and $j' \in \{0,1,2,3\}$. These paths build tree $T_{(i,2)}$.

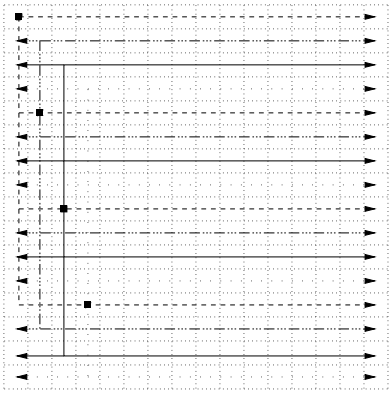

Figure 4.6: Trees in a partitioning of $M_{3,16}$.

This figure shows all connections paths in the submesh. These paths build trees $T_{(i,0)}, \ldots, T_{(i,3)}$.

6. Let $q = (q_0, \ldots, q_{r'-2}) \in [n]^{r'-1}$. For $i \in [n]^{r'}$, $j \in [s]$, we set $T_{(i,j)} = (V_{(i,j)}, E_{(i,j)}, exit_{(i,j)})$. We define $T_{(i,j)}$ as union of directed paths from $exit_{(i,j)}$ to $entry_{(i',j')}(i,j)$, $(i',j') \in [n]^{r'} \times [s]$. The root $exit_{(i,j)}$ of $T_{(i,j)}$ is the node $(i, jm, j, q) \in [n]^r$. For $i, i' \in [n]^{r'}$, $j, j' \in [s]$ the path from $exit_{(i,j)}$ to $entry_{(i',j')}(i,j)$, where $i' = (i'_0, \ldots, i'_{r'-1})$, is given as follows

$$\begin{aligned}
exit_{(i,j)} &= \\
(i, jm, j, q_0, \ldots, q_{r'-2}) &\Longrightarrow^* \\
(i, j'm + j, j, q_0, \ldots, q_{r'-2}) &\Longrightarrow^* \\
(i, j'm + j, i'_0, q_0, q_1 \ldots, q_{r'-2}) &\Longrightarrow^* \\
(i, j'm + j, i'_0, i'_1, q_1 \ldots, q_{r'-2}) &\Longrightarrow^* \\
\vdots &\Longrightarrow^* \\
(i, j'm + j, i'_0, \ldots, i'_{r'-2}, q_{r'-2}) &\Longrightarrow^* \\
(i, j'm + j, i'_0, \ldots, i'_{r'-1}) &= \\
entry_{(i',j')}((i,j)).
\end{aligned}$$

Let $I = [n]^{r'} \times [s]$ and $q_0 = \cdots = q_{r'-2} = 0$. Then

$$((S_{(i,j)})_{(i,j)\in I}, (D_{(i,j)})_{(i,j)\in I}, (entry_{(i,j)})_{(i,j)\in I}, (T_{(i,j)})_{(i,j)\in I})$$

is an $(sn^{r'}, sn^{r'})$-partitioning of $M_{r,n}$ of size

$$(mn^{r'} + (n \bmod m) \cdot n^{r'}, mn^{r'} + (n \bmod m) \cdot n^{r'}, (s-1)m + (n-1)^{r'}).$$

Combining the results for even and odd $r > 1$ we get the following theorem.

Theorem 4.5

Algorithm **KKOblivious** *is a k-k routing algorithm on $M_{r,n}$, $r > 1$, with a running time of $O(kn^{\frac{r}{2}})$ steps and a buffer size of $O(k)$. The running time matches the lower bound of Theorem 4.1 asymptotically.*

If an additional information field of size $O(\log k + \log n)$ is allowed, then the buffer size can be bounded by $k+9$. If an additional information field of size $O(\log n)$ is allowed, then the buffer size can be bounded by $2k+8$.

Proof:

For the buffer size see the discussion at the end of the previous section and note that the trees in the partitionings are node disjoint. ◇

4.5 Conclusion.

In this chapter we presented an oblivious k-k routing algorithm with an asymptotically optimal running time and buffer size $O(k)$ for r-dimensional meshes and networks $\mathcal{N}$ for which a (t_1, t_2)-partitioning of size $(O(|V_{\mathcal{N}}|^{\frac{1}{2}}), O(|V_{\mathcal{N}}|^{\frac{1}{2}}), O(|V_{\mathcal{N}}|^{\frac{1}{2}}))$ exists. Although the algorithm has a small buffer size and an asymptotically optimal running time there remain open problems.

- **simple paths, shortest paths:** One important idea leading to our algorithm is counting. In the source blocks, we count the number of packets destined for a destination block and in the destination blocks, we count values for dynamic routing. This counting is done on cycles. Thus, the path of a packet is not simple and hence is not a shortest path. We do not know whether it is possible to design a k-k routing algorithm for $M_{r,n}$ with a running of $O(kn^{\frac{r}{2}})$ and buffer size $O(k)$ that uses shortest paths.

- **constants:** The constants of our algorithm are large. For example, on $M_{2,n}$ we need $50n$ steps and a buffer size of 10. The case $r = 2$ and possibly the case $r = 3$ is of practical interest. Hence a reduction of the constants for these cases is of interest.

- **number of bits used:** The number of bits used on a processor for calculations is large. Hence a reduction of this number is of interest.

- **complexity of calculation:** The calculation of η is complex. Possibly the dynamic routing problem can be solved efficiently without calculating η. Here further research is needed.

- **buffer size:** It would be interesting to know how far the buffer size can be reduced. For example: Is it possible to design an oblivious permutation routing algorithm for $M_{r,n}$ with a running time $O(n^{\frac{r}{2}})$ and buffer size 1?

Chapter 5

OTIS Networks.

This chapter investigates several aspects of routing on OTIS networks. In particular, we show that for any OTIS-G network of fixed degree an oblivious k-k routing algorithm with an asymptotically optimal running time and buffer size $O(k)$ exists. We give a k-k sorting algorithm for the OTIS-Mesh whose running time comes close to the bisection and diameter bound. We extend the definition of OTIS-G networks and achieve a reduction of the diameter by a factor of approximately two for several networks G.

5.1 Introduction.

In the computing community there is a growing interest in optics. Optical interconnections are an interesting alternative to electronic interconnections. They provide high interconnectivity and large bandwidth. So optical networks have gained much attention in our days. However, electronic interconnections have advantages, too. They perform better for small distances [6]. The Optical Transpose Interconnection System (OTIS) proposed in [35] defines networks in which optical and electronical links are used.

The processors in an OTIS network are partitioned in groups where the connections within the groups are realized by electronic links and the connections among the groups are realized by optical links. Observations in [33] have shown that it is favorable to choose the size of the groups equal to the number of groups. Afterwards only OTIS networks were considered where the number of groups are equal to the number of processors in the groups [58, 52, 51, 53, 39, 34]. Here we also restrict our attention to this case. In an OTIS network a processor p of group g is connected via an optical link to processor g of group p. The electronical connections within the groups are given by the topology of the group, where the group can be any (connected) graph G. An OTIS network, where the groups are isomorphic to a graph G is called an *OTIS-G network*.

Several parallel algorithms have been developed for different OTIS-G networks recently, e.g. algorithms for routing [5, 50, 52, 42], sorting [42, 39], selection [42], data movement [51], matrix multiplication [54], and image processing [55]. The majority of these algorithms have been designed for the OTIS-$M_{2,n}$ network [50, 42, 51, 54, 55]. Such a network is also called an OTIS-Mesh. We consider the k-k sorting problem on the OTIS-Mesh and give an algorithm with buffer size $k+4$ whose running time comes close to the bisection and diameter bound. Up to now the k-k sorting problem was not considered on the OTIS-Mesh[1] or other OTIS-G networks. In addition to k-k sorting on the OTIS-Mesh, oblivious routing on OTIS-G networks is considered. The results of the previous chapter are applied and A k-k routing algorithm for OTIS-G networks, where G is a graph of fixed degree, with an asymptotically optimal running time and $O(k)$ buffer size is achieved.

A lower bound for many basic parallel problems, including routing and sorting problems, is the diameter of the network. Hence reducing the diameter

[1] In [39], we have presented k-k sorting on the OTIS-Mesh.

gives the possibility to design faster algorithms for these problems. The diameter of OTIS-G networks is reduced by adding at most one (optical) link to each processor g in group g. These additional links do not increase the degree of the network. The resulting networks are called *Extended OTIS-G networks*. Several graphs G are investigated: hypercubes, one- and two-dimensional meshes and rings. It is shown that the extension is optimal in the following sense. Each addition of links to processors g in groups g in an OTIS-G network that does not increase the degree of the network results in a diameter of equal or greater size. Furthermore, if G is regular, then any addition of links to the OTIS-G network that does not increase the degree of the network results in a diameter of equal or greater size.

The rest of this chapter is organized as follows. In the next Section the definition of OTIS-G networks is given. A lower bound for routing in OTIS-G networks is presented in Section 5.3. The k-k sorting problem on the OTIS-Mesh in ivestigated in Section 5.4. Oblivious routing is considered in Section 5.5. Finally, the diameter of OTIS-G and Extended OTIS-G networks is calculated in Section 5.6 and Section 5.7.

5.2 Definition of OTIS-G Networks.

We begin with the definition of OTIS-G networks.

Definition 5.1 (OTIS-G) *Let $G = (V, E)$, $|V| > 1$, be a graph. The OTIS-G network (or O_G network for short) is a graph with node set $V_G \times V_G$ and edge set $E_o \cup E_e$, where*

$$E_e = \{\{(g,p),(g,p')\} \mid g,p,p' \in V \wedge \{p,p'\} \in E\}$$

is the set of intra-group or electronic links and

$$E_o = \{\{(g,p),(p,g)\} \mid p,g \in V, p \neq g\}$$

is the set of inter-group or optical links. We use $\longleftrightarrow_o$ *to denote optical links and* $\longleftrightarrow_e$ *to denote electronical links.*

In the case that G is a two-dimensional mesh (a ring, a hypercube) we call OTIS-G an OTIS-Mesh (OTIS-Ring, OTIS-Hypercube).

For all $g \in V_G$ *we call the graph* G_g *with node set* $\{g\} \times V_G$ *and edge set*

$$\{\{(g,p),(g,p')\} \mid p,p' \in V, \{p,p'\} \in E_G\}$$

group g of OTIS-G.

It follows directly from the definition of OTIS-G networks that each of its $|V_G|$ groups is isomorphic to G. If $\{G_n \mid n \in \mathbb{N}\}$ is a family of graphs of fixed degree, $\{$ OTIS-$G_n \mid n \in \mathbb{N}\}$ is a family of fixed degree.

Figure 5.1 presents an OTIS-$M_{2,2}$ network. The network consists of groups $(0,0)$, $(0,1)$, $(1,0)$, and $(1,1)$. The intra-group links are shown as solid lines and the inter-group links are shown as dashed lines.

5.3 A Lower Bound for Routing on OTIS-G Networks.

We prove a lower bound for k-k routing[2] on an OTIS-G network. The proof is based on a bisection and diameter argument. Note that the following lower bound is independent of the bandwidth of optical links.

Theorem 5.2 *Any k-k routing (sorting) algorithm on an OTIS-G network requires at least* $\max\{2D(G)+1, \frac{k}{bw(G)}\}$ *steps in the worst case.*

[2] and k-k sorting

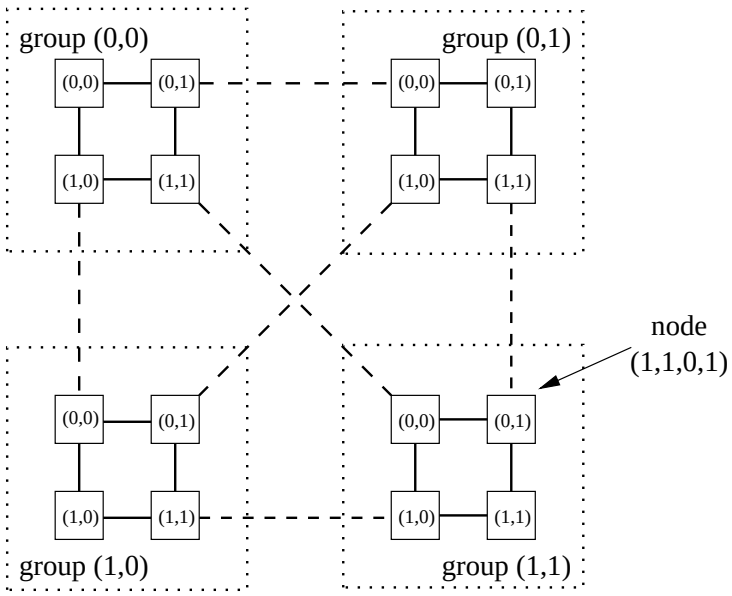

Figure 5.1: The structure of OTIS-$M_{2,2}$.

This figure shows intra-group links (electronical links) of OTIS-$M_{2,2}$ as solid lines and inter-group links (optical links) as dashed lines.

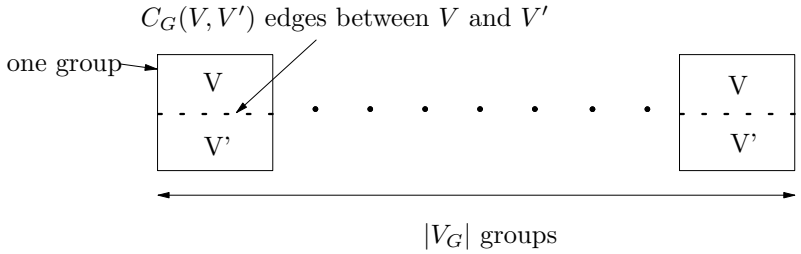

Figure 5.2: An OTIS-G network divided in two areas.

Proof:

By Theorem 5.18 the diameter of an OTIS-G network is $2D(G) + 1$.

Let $V \subseteq V_G$ such that $\frac{C_G(V,V_G-V)}{|V|} = bw(G)$ and $|V| = \left\lfloor \frac{|V_G|}{2} \right\rfloor$. Let $V' = V_G - V$ and $V_1 = (V_G \times V_G) - ((V \times V) \cup (V' \times V'))$. See Figure 5.2. Then

- $C_{O_G}(V_1, (V_G \times V_G) - V_1) = |V_G| C_G(V, V')$ and
- $|V_1| = \left\lfloor \frac{|V_G|^2}{2} \right\rfloor$.

We have $\left\lfloor \frac{|V_G|^2}{2} \right\rfloor = |V| \cdot |V_G|$. Therefore

$$bw(O_G) \leq \frac{|V_G| C_G(V,V')}{\left\lfloor \frac{|V_G|^2}{2} \right\rfloor} = \frac{C_G(V,V')}{|V|} = bw(G).$$

Note that no optical link connects a node of set V_1 with a node of set $V_G \times V_G - V_1$. Hence the lower bound is independent of the bandwidth of optical links. ◇

5.4 Sorting on the OTIS-Mesh.

In this section we discuss the *k-k sorting problem* on OTIS-$M_{2,n}$ networks[3].

[3]We present similar results in [39].

5.4.1 Preliminaries

An OTIS-$M_{2,n}$ network consists of n^4 nodes. The n^4 nodes are grouped in n^2 two-dimensional meshes of side length n. The connections between the n^2 meshes are built according to the OTIS law: A node p in mesh g is connected to node g in mesh p, $g, p \in [n]^2$. If we place the n^2 meshes as a two-dimensional mesh of side length n (see Figure 5.1) a node $((g_r, g_c), (p_r, p_c))$ of the OTIS-$M_{2,n}$ network lies in the mesh in row g_r and column g_c. Within this mesh the node lies in row p_r and column p_c, $g_r, g_c, p_r, p_c \in [n]$. Instead of $((g_r, g_c), (p_r, p_c))$ we simply write (g_r, g_c, p_r, p_c).

Sorting problems are similar to routing problems. In a k-k sorting problem each processor is source and destination of exactly k packets, where each packet contains a key (in the message field) drawn from a totally ordered set. The packets are assumed to lie in k *layers* within the processors. A pair consisting of a processor and a layer is called a *place*. An *indexing* of places is a bijection $\mathcal{I} : [n]^4 \times [k] \longrightarrow [kn^4]$. The goal in k-k sorting is to arrange the packets such that the packet containing the i-th smallest key is moved to the place with index $i-1$. Analogously to routing, a 1-1 sorting problem is called a *permutation sorting problem*.

For general k, the k-k sorting problem was not considered so far by other authors on the OTIS-Mesh. For permutation sorting Sahni and Wang presented in [51] a deterministic algorithm with running time $11n+o(n)$ without considering its buffer size and Rajasekaran and Sahni presented in [42] a randomized algorithm with running time $8n + o(n)$ and buffer size $O(1)$. We present a k-k sorting algorithm with running time $\max\{8n + o(n), 2kn + o(kn)\}$ and buffer size $k + 4$.

We solve the sorting problem by using sorting with all-to-all mappings. In the next section, we describe how to sort with all-to-all mappings. After-

wards, we present an embedding of $M_{4,n}$ in the OTIS-$M_{2,n}$ network, give basic definitions and notations, present the all-to-all mapping and the sorting algorithm. Finally, we give a lower bound for routing and sorting on OTIS-$M_{2,n}$ networks and conclude with a comparison of the sorting algorithm with algortihms designed for meshes.

For the rest of the section we choose $\varepsilon \in \mathbb{R}$ such that $0 < \varepsilon < 1$ and assume that $n^{\varepsilon} \in \mathbb{N}$.

5.4.2 Sorting with All-to-All Mappings.

The aim of this section is to describe how to sort in a network with the help of an all-to-all mapping that distributes data uniformly within the network. This sorting method is well known and was introduced in [25]. A similarity between sorting with all-to-all mappings and Leighton's Columnsort [30] exists. We repeat the ideas of sorting with all-to-all mappings, give the results, explain their correctness and present their realization in the OTIS-Mesh. Finally, we shortly discuss how sorting based on all-to-all mappings can be used to solve routing problems.

For sorting with all-to-all mappings, we divide the network into l blocks of equal size and each block into l subblocks of equal size called *bricks*. Hence there are l blocks of size $\frac{|V_{\mathcal{N}}|}{l}$ and l^2 bricks of size $\frac{|V_{\mathcal{N}}|}{l^2}$ in the network (we assume that $l^2||V_{\mathcal{N}}|$). The blocks have an index from 0 to $l-1$ and the bricks in each block have an index from 0 to $l-1$. We choose the indexing of the places such that all indices in block i are smaller than the smallest index in block $i+1$ for $i \in [l-1]$. We want to solve a k-k sorting problem by using sorting with all-to-all mappings.

Roughly speaking sorting with all-to-all mappings consists of the following five steps.

1. Sort each block.
2. Perform an all-to-all mapping.
3. Sort each block.
4. Perform an all-to-all mapping.
5. Sort all pairs of blocks.

The correctness of the sorting method follows from the 0-1 principle [20].

In the first step, the blocks are sorted such that the brick with index i (called brick i) in a block gets all packets containing a key with rank $\equiv i \pmod{l}$. So, after step 1, the number of ones in any two bricks i, i' within a block differs at most by one. In the second step, from every block the contents of one brick, i.e., the packets in the nodes of the brick, is sent to every block. To be more precise, if $ata_{1,i} : [l] \longrightarrow [l]$ describes this first transport, i.e., $ata_{1,i}(j) = k$ if the contents of brick j in block i is transported to block k (we omit to describe where the contents of brick j is transported to in block k), then for all $i \in [l]$ the mapping $ata_{1,i}$ is bijective. After the transport, the number of ones in any two blocks of the network differs at most by l. In the third step of the method, the blocks are sorted such that the keys of all packets in brick i of the block are smaller than the smallest key of a packet in brick $i+1$ of the block, $i \in [l]$. So at most one brick in a block contains zeros and ones. We call a brick that contains zeros and ones *dirty*. Assume that all bricks contain the same number of packets and assume further that the number of packets is at least l. Remember that the number of ones in any two blocks differs at most by l after step 2 of the sorting method. Therefore, if in blocks i and i' dirty bricks with index d_i and $d_{i'}$ exist, then the above condition on the number of packets in a brick implies $|d_i - d_{i'}| \leq 1$. So the index of the dirty brick in the blocks differ at most by one. In the fourth

step, the all-to-all mapping transports the contents of the brick j in block i to block j, i.e. $ata_{2,i}(j) = j$ for all $i, j \in [l]$. After the second all-to-all mapping at most two blocks contain dirty bricks and these blocks have index i and $i+1$ for an $i \in [l-1]$. Hence step 5 concludes the sorting.

The running time of a k-k sorting algorithm based on sorting with all-to-all mappings depends on the running time for sorting the blocks in steps 1, 3, and 5 and the running time for performing the two all-to-all mappings. A lower bound for routing an all-to-all mapping is given by the diameter and halve the bisection bound. The running time for sorting the blocks in step 1, 3, and 5 depends on the size of the blocks and the maximal number of packets in a processor at the beginning of the sorting. The running time for step 5, in which pairs of blocks are sorted, additionally depends on the distance between two blocks of a pair. To bound the maximal number of packets in a processor in the beginning of step 3 and step 5, we need that the all-to-all mappings in step 2 and step 4 are designed such that each brick in a block receives the contents of one brick.

As seen in the above discussion, the sorting method only works correctly if any brick contains at least l packets. In the case of k-k sorting on a network $\mathcal{N}$, this implies

$$l^3 \leq k|V_{\mathcal{N}}|. \tag{5.1}$$

In the case of the OTIS-Mesh, it is possible to embed $M_{4,n}$ in OTIS-$M_{2,n}$ with constant dilation, constant congestion and load 1. So every algorithm for $M_{4,n}$ can be performed with a constant slowdown on OTIS-$M_{2,4}$. For r-dimensional meshes of side length n, inequality 5.1 results in blocks with at least $\frac{n^{2r/3}}{k^{1/3}}$ nodes. Hence we are able to choose blocks of side length n^ϵ, $\frac{2}{3} \leq \epsilon < 1$. Thus, step 1 and step 3 can be done in $O(kn^\epsilon)$ steps. Furthermore, in r-dimensional meshes a *continuous indexing* of the blocks can be used. In a continuous indexing of blocks, any two blocks with index

i and $i+1$ are neighbouring. Therefore, step 5 can be done in $O(kn^{\epsilon})$ steps. The diameter bound for an r-dimensional mesh is $r(n-1)$, and the bisection bound is $\frac{kn}{2}$. Therefore, the running time is dominated by the time required to perform the all-to-all mappings ([24, 25, 26, 27, 28]).

Implementing the all-to-all based k-k sorting for $M_{4,n}$ on OTIS-$M_{2,n}$, using the embedding results in an asymptotically optimal running time of the sorting algorithm. However, the constants of the running time of such an algorithm are very large. We use the embedding in step 1, 3, and 5 to sort the blocks with a running time $O(kn^{\varepsilon})$, $\varepsilon < 1$. For the OTIS-Mesh the main task is to find an efficient algorithm for the all-to-all mapping.

Routing with Sorting. Sorting with all-to-all mappings can be applied to solve routing problems. In [25, 27, 28] a detailed description can be found. Obviously a full k-k routing problem can be solved by solving a k-k sorting problem. A problem occurs when there are less than $k|V_{\mathcal{N}}|$ packets in the network. In this case, instead of sorting the blocks in the third step, we route a packet in block j destined for block i to brick i in block j. It is easy to see that this routing sends at most $\frac{k|V_{\mathcal{N}}|}{l^2} + l$ packets to a brick in a block. Hence we have to move at most l packets destined for brick i to brick $i-1$ or brick $i+1$. In the case of r-dimensional meshes this additional routing in the blocks can be done efficiently ([25, 27, 28]).

5.4.3 Emulation of $M_{4,n}$ by OTIS-$M_{2,n}$.

An emulation of a four-dimensional mesh by an OTIS-Mesh was first described in [58]. The emulation uses an embedding of $M_{4,n}$ into the OTIS-$M_{2,n}$ network. The embedding maps node (i,j,k,l) of $M_{4,n}$ to node (i,j,k,l) of OTIS-$M_{2,n}$, $i,j,k,l \in [n]$. The edges of $M_{4,n}$ are mapped to paths in the OTIS-$M_{2,n}$ network in the following way ([58, 39]):

1.) $i, j, l \in [n]$, $k \in [n-1]$:

$$\{(i,j,k,l),(i,j,k+1,l)\}$$
$$\mapsto$$
$$(i,j,k,l) \longleftrightarrow_e (i,j,k+1,l)$$

2.) $i, j, k \in [n]$, $l \in [n-1]$:

$$\{(i,j,k,l),(i,j,k,l+1)\}$$
$$\mapsto$$
$$(i,j,k,l) \longleftrightarrow_e (i,j,k,l+1)$$

3.) $j, k, l \in [n]$, $i \in [n-1]$:

$$\{(i,j,k,l),(i+1,j,k,l)\}$$
$$\mapsto$$
$$(i,j,k,l) \longleftrightarrow_o (k,l,i,j) \longleftrightarrow_e (k,l,i+1,j) \longleftrightarrow_o (i+1,j,k,l)$$

4.) $i, k, l \in [n]$, $j \in [n-1]$:

$$\{(i,j,k,l),(i,j+1,k,l)\}$$
$$\mapsto$$
$$(i,j,k,l) \longleftrightarrow_o (k,l,i,j) \longleftrightarrow_e (k,l,i,j+1) \longleftrightarrow_o (i,j+1,k,l)$$

5.) If $(i,j) = (k,l)$ the first optical link in 3.) and 4.) is omitted.

6.) If $(i+1,j) = (k,l)$ the second optical link in 3.) is omitted and if $(i,j+1) = (k,l)$ the second optical link in 4.) is omitted.

This describes an embedding of $M_{4,n}$ into an OTIS-$M_{2,n}$ network with dilation 3, congestion 8, and load 1.

With the help of the embedding it is possible to emulate $M_{4,n}$ with constant slowdown and at most four additional buffers.

Theorem 5.3 *Any algorithm that needs T steps and a buffer size of B on $M_{4,n}$ can be performed in $14T$ steps with a buffer size of $B+4$ on an OTIS-$M_{2,n}$ network.*

Proof:

We use the embedding described above and a step by step simulation. A node (i,j,k,l) in the OTIS-Mesh simulates node (i,j,k,l) of the four-dimensional mesh. To simulate one communication step the following simple communication schedule consisting of fourteen steps is used. All packets that have to use a path of 1.) or 2.) are sent in the first step. These packets reach their destination after one step. There are at most four packets in a node that have to use a path of 3.), 4.), 5.), or 6.). Packets that have to use a path of 3.), 4.), or 6.) are sent in step 2, one in step 5, one in step 8, and one in step 11. Packets that have to use a path of 5.) are sent in step 3, step 6, and so on. A fixed order, which is the same on all nodes, is used to send the packets, i.e., all processors (i,j,k,l) send a packets to neighbour $(i+1,j,k,l)$ in step 2 (3), to neighbour $(i-1,j,k,l)$ in step 5 (6) and so on. For the traveling packets at most one additional buffer is reqired on a processor after step 2. Now we consider a node (i,j,k,l), where $(i,j) \neq (k,l)$, i.e. packets on this node do not use a path of 5.). After step 1, at most four packets on this node have to be sent, after step 2 at most 3, after step 5 at most 2, and so on. So an additional buffer size of at most four is needed. On a node (i,j,k,l), where $(i,j) = (k,l)$, there are at most four packets after step 2, three packets after step 3, and so on. In step 3, nodes receive packets via an optical links. So node (i,j,i,j) does not receive a packet. Therefore, an additional buffer size of at most four is needed. The last packet reaches its destination in step fourteen. $\diamond$

If we assume a bandwidth of four for the optical links in the OTIS-Mesh, then we can achieve a faster simulation of the four-dimensional mesh.

Theorem 5.4 *If every optical link has a bandwidth of at least four, then any algorithm that needs T steps and a buffer size of B on $M_{4,n}$ can be performed in $3T$ steps with a buffer size of $B+4$ on an OTIS-$M_{2,n}$ network.*

Proof:

We use the embedding described above and a step by step simulation of the network. We need three steps. All packets that use paths of 1.) or 2.) use an electronical link in the first step and reach their destination after one step. For these packets no additional buffer size is needed. All other packets have to travel a distance of at least two and at most three and have to use up two optical links, one in step 1 and one step 3. Packets of 5.) do not travel in step 1 and packets of 6.) do not travel in step 3.

In step 2, each processor receives at most 4 packets via an optical link. Note that a node (i, j, i, j) is not able to send a packet via an optical link in step 1 but it also receives no packets via an optical link in step 2. The packets that reach a processor via an optical link in step 2 have to leave it via different electronical links. There are four electronical links and so every packet is able to leave the node in step 2. Thus at most four additional buffers are needed in step 2.

In step 3, all packets reach a node via an electronical link. At most four packets reach a node (i, j, k, l). If $(i, j) = (k, l)$, then the processor (i, j, k, l) is destination of these packets. Otherwise, the packets are sent to their destination via an optical link. Hence an additional buffer size of at most 4 is required. ◇

5.4.4 Substructures in the OTIS-Mesh.

In the following, we define *blocks*, *superblocks*, and *bricks*. A block is the realization of a four-dimensional submesh of $M_{4,n}$ in the OTIS-$M_{2,n}$ network.

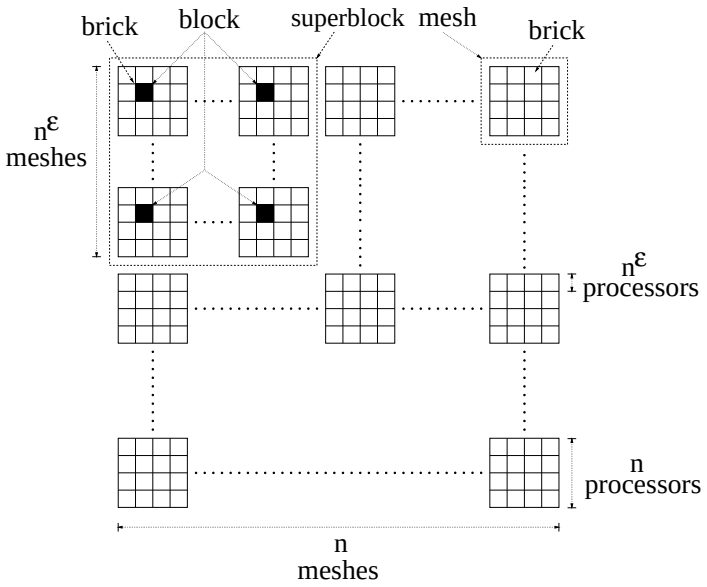

Figure 5.3: Superblocks, blocks, and bricks in an OTIS-Mesh network.

A superblock is a collection of blocks and a brick is the cut of a block and a mesh. We give the formal definitions.

For $a, b \in [n]$ we write $M(a, b)$ for the group (a, b) of the OTIS-$M_{2,n}$ network. For $a, b, c, d \in [n^{1-\varepsilon}]$ the *block* $B(a, b, c, d)$ of side length n^ε consists of processors (i, j, k, l), where $an^\varepsilon \leq i < (a+1)n^\varepsilon$, $bn^\varepsilon \leq j < (b+1)n^\varepsilon$, $cn^\varepsilon \leq k < (c+1)n^\varepsilon$, and $dn^\varepsilon \leq l < (d+1)n^\varepsilon$. For $a, b \in [n^{1-\varepsilon}]$ the *superblock* $SB(a, b)$ consists of blocks $B(a, b, c, d)$, where $c, d \in [n^{1-\varepsilon}]$. If the cut of block $B(a, b, c, d)$ and mesh $M(i, j)$ is not empty, we call it *brick* of block $B(a, b, c, d)$ and denote it by $< a, b, c, d, i, j >$. The cut of block $B(a, b, c, d)$ and mesh $M(i, j)$ is not empty if and only if $an^\varepsilon \leq i < (a+1)n^\varepsilon$ and $bn^\varepsilon \leq j < (b+1)n^\varepsilon$. Figure 5.3 shows meshes, blocks, superblocks, and bricks. The filled black boxes build one block. Each of these filled boxes is one brick of the block.

There are $n^{2(1-\varepsilon)}$ superblocks, $n^{4(1-\varepsilon)}$ blocks, and $n^{4(1-\varepsilon)}n^{2\varepsilon}$ bricks in an OTIS-$M_{2,n}$ network. Each superblock consists of $n^{2(1-\varepsilon)}$ blocks and each block consists of $n^{2\varepsilon}$ bricks. To describe the all-to-all mapping we need some indexings. We introduce indexings for meshes, blocks, superblocks, bricks, and processors. We use row-major indexing for most of the structures: The processors in a brick, the bricks in a block, the blocks in a superblock, and the superblocks within an OTIS-Mesh.

In Section 5.4.5 we use a blocked row-major indexing of the blocks in the OTIS-Mesh, i.e., the index j of a block B in the OTIS-Mesh can be written as $j = j_1 n^{2(1-\varepsilon)} + j_2$, where j_1 is the index of the superblock SB in which B lies and j_2 is the index of B in the superblock. For example, the index of block $B(a, b, c, d)$ in the OTIS-Mesh is $(an^{1-\varepsilon} + b)n^{2(1-\varepsilon)} + cn^{1-\varepsilon} + d$. For sorting with all-to-all mappings (Section 5.4.6) we need a different indexing of the blocks in the OTIS-Mesh. Each block of the OTIS-$M_{2,n}$ network can be seen as a block in $M_{4,n}$. Since we use an embedding of $M_{4,n}$ into the OTIS-Mesh to sort the OTIS-Mesh, we need a continuous indexing of the blocks in $M_{4,n}$. In Section 5.4.6 we use an arbitrary continuous index of the blocks in $M_{4,n}$.

Now we give the definitions of the indexings. Brick $< a, b, c, d, e, f >$ has index $(e - an^{\varepsilon})n^{\varepsilon} + (f - bn^{\varepsilon})$ in block $B(a, b, c, d)$. Block $B(a, b, c, d)$ has index $cn^{1-\varepsilon} + d$ in superblock $SB(a, b)$. If a block B in superblock SB has index x and brick BR in B has index y, then we call BR as brick x in block y. Processor (e, f, g, h) has index $(e-a)n^{\varepsilon}+(f-b)$ in brick $< a, b, c, d, e, f >$. If a processor P in brick BR has index z, we call P processor z in brick x.

5.4.5 All-to-All Mapping.

This section describes how an all-to-all mapping can be implemented on the OTIS-Mesh. To achieve our target we first perform a k-k routing of the

packets in the meshes to distribute the contents of the bricks within the superblock. Afterwards we use the optical links. This step distributes the packets of a brick to the bricks of one block. Then we perform a k-k routing in the blocks to collect these packets in one brick of the block. Finally, we perform a k-k routing in the meshes to distribute the bricks to the blocks in their destination superblock.

Algorithm Brick Transport.

1. Within all superblocks do: Move the packets from brick j in block i to brick j in block $(i + \lfloor \frac{j}{n^\varepsilon} \rfloor) \bmod n^{2(1-\varepsilon)}$, $0 \leq j < n^{2\varepsilon}$, $0 \leq i < n^{2(1-\varepsilon)}$.

2. For all processors do: Use the optical links to move all packets from processor (e, f, g, h) to processor (g, h, e, f).

3. Within all blocks do: Move the packets from processor j in brick k to processor k in brick j, $0 \leq j, k < n^{2\varepsilon}$

4. Within all superblocks do: Move the packets from brick j in block i to brick j in block $(i + (j \bmod n^\varepsilon)) \bmod n^{2(1-\varepsilon)}$, $0 \leq j < n^{2\varepsilon}$, $0 \leq i < n^{2(1-\varepsilon)}$.

Theorem 5.5 (Running Time of Algorithm Brick Transport.)
If every processor holds at most k packets, then algorithm **Brick Transport** *can be performed with buffer size $k+4$ in $\max\{4n + o(n), kn + o(kn)\}$ steps.*

Proof:
A superblock consists of $n^{2\varepsilon}$ meshes. These meshes build n^ε rows (columns) of meshes (see Figure 5.3). Each row (column) consists of n^ε meshes. We number these rows (columns) from 0 to $n^\varepsilon - 1$.

We consider a mesh in row x and column y of meshes in a superblock. It contains $n^{2(1-\varepsilon)}$ bricks from $n^{2(1-\varepsilon)}$ blocks. Each brick in the mesh has index

$j = xn^\varepsilon + y$ within its block. Step 1 does not change the index of a brick. Hence step 1 moves the packets within a mesh.

The bricks in a mesh build $n^{1-\varepsilon}$ rows (columns) of bricks. Each such row (column) contains $n^{1-\varepsilon}$ bricks. A brick in row x' and column y' of bricks in the mesh belongs to a block with index $x'n^{1-\varepsilon} + y'$ in the superblock. These block indices of the bricks within a mesh yields a row-major indexing. Step 1 shifts the packets by x bricks with respect to this indexing, i.e., the packets are shifted from a brick with (block) index $x'n^\varepsilon + y'$ and to a brick with (block) index $x'n^{1-\varepsilon} + y' + x \bmod n^{2\varepsilon}$. Therefore, step 1 is a k-k routing in a mesh. Furthermore, step 1 transports any two bricks in a row (column) of bricks to different columns (rows) of bricks. Hence it can be done by first performing a k-k routing in the rows (columns) and then performing a k-k routing in the columns (rows). Both, the first and the second routing, have a very simple structure. Assume that we first route within the rows. Then all packets moving to the right (left) in the first routing have to travel the same distance. The distance is $(x \bmod n^{1-\varepsilon})n^\varepsilon$ (left: $(n-(x \bmod n^{1-\varepsilon})n^\varepsilon)$). Note that the distance is the same for all meshes in a column of meshes. All packets moving up (down) in the second routing have to travel the same distance. The distance is $n^\varepsilon \left\lfloor \frac{x}{n^{1-\varepsilon}} \right\rfloor$ (down: $(n - n^\varepsilon \left\lfloor \frac{x}{n^{1-\varepsilon}} \right\rfloor)$). Again the distance is the same for all meshes in a row of meshes. We use the technique of packet colouring to halve the running time. In a mesh $M(a,b)$ we colour $\left\lceil \frac{k}{2} \right\rceil$ packets on processor (a,b,c,d) black and $\left\lfloor \frac{k}{2} \right\rfloor$ white if and only if $c+d$ is even, and $\left\lceil \frac{k}{2} \right\rceil$ packets white and $\left\lfloor \frac{k}{2} \right\rfloor$ packets black else. We route all black packets first within the rows and then within the columns and all white packets first within the columns and then within the rows.

We use the farthest destination first queuing strategy to route the packets. So step 1 can be done with a buffer size of at most $k+4$ in $2n + O(1)$ steps for $k \leq 4$ and in $\frac{kn}{2} + o(kn)$ steps for $k > 4$ (e.g. see Lemma 1 in [19]).

Step 4 can be done analogously to step 1. Step 4 shifts the packets by y bricks with respect to the (block) indexing of the bricks in a mesh. Hence step 4 can be done with a buffer size of at most $k+4$ in $2n+O(1)$ steps for $k \leq 4$ and in $\frac{kn}{2} + o(kn)$ steps for $k > 4$.

Step 3 is a k-k routing in a block. By Theorem 5.3 and [25] step 3 can be performed in $o(kn)$ steps with a buffer size of at most $k+4$.

Step 2 needs k steps and a buffer size of k. ◇

In the following we observe where the packets of a brick are moved to by algorithm **Transport Bricks**.

Lemma 5.6 *Algorithm* **Brick Transport** *moves the packets from brick* $< a,b,c,d,e,f >$, $a,b,c,d \in [n^{1-\varepsilon}]$, $an^\varepsilon \leq e < (a+1)n^\varepsilon$, $bn^\varepsilon \leq f < (b+1)n^\varepsilon$, *to brick* $< a',b',c',d',e',f' >$, *where*

$$\begin{aligned}
a' &= \left\lfloor \tfrac{z}{n^{1-\varepsilon}} \right\rfloor \\
b' &= z \bmod n^{1-\varepsilon} \\
c' &= \left\lfloor \tfrac{z'}{n^{1-\varepsilon}} \right\rfloor \\
d' &= z' \bmod n^{1-\varepsilon} \\
e' &= e + (c'-a)n^\varepsilon \\
f' &= f + (d'-b)n^\varepsilon \\
z &= (cn^{1-\varepsilon} + d + e - an^\varepsilon) \bmod n^{2(1-\varepsilon)} \\
z' &= (an^{1-\varepsilon} + b + f + (d'-2b)n^\varepsilon) \bmod n^{2(1-\varepsilon)}.
\end{aligned}$$

Proof:

Let $< a,b,c,d,e,f >$ be a brick. We use the definition of a',b',c',d',e',f',z, and z' as given in the lemma.

Step 1 moves the packets of $< a,b,c,d,e,f >$ to brick $< a,b,c',d',e,f >$. Step 2 and step 3 moves them to brick $< c',d',a,b,e+(c'-a)n^\varepsilon, f+(d'-b)n^\varepsilon >$ and step 4 moves them to brick $< a',b',c',d',e',f' >$.

Remark: Step 2 in combination with step 3 moves brick $< a, b, c, d, e, f >$ to brick $< c, d, a, b, e + (c - a)n^{\varepsilon}, f + (d - b)n^{\varepsilon} >$. Thus these steps define a permutation on the set of bricks in the OTIS-Mesh. $\diamond$

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

From the above observation we know that algorithm **Transport Bricks** moves the packets from a brick to another brick. Let $\mathcal{B}$ be the set $\{(B, BR) \mid B$ is a block in the OTIS-Mesh and BR is a brick in $B\}$ and let $ata : \mathcal{B} \longrightarrow \mathcal{B}$, where $ata(B, BR) = (B', BR')$ if and only if algorithm **Transport Bricks** moves the packets from brick BR in block B to brick BR' in block B'.

We have to show that each brick of a block receives the contents of one brick and that each block receives one brick from every block, i.e., we have to show that ata is bijective and for all blocks B, B' in the OTIS-Mesh

$$|ata(\{(B, BR) \mid BR \text{ is brick in } B\}) \cap \{(B', BR) \mid BR \text{ is brick in } B'\}| = 1$$

holds.

Theorem 5.7 *For* $\varepsilon = \frac{2}{3}$ **Brick Transport** *realizes an all-to-all mapping, i.e., ata is bijective and for all* $B, B' \in \mathcal{B}$

$$|ata(\{(B, BR) \mid BR \textit{ is brick in } B\}) \cap \{(B', BR) \mid BR \textit{ is brick in } B'\}| = 1.$$

Proof:

We consider block $B(a, b, c, d)$. First we observe that each block in the OTIS-Mesh gets packets from exactly one brick of block $B(a, b, c, d)$.

Block $B(a, b, c, d)$ consists of bricks $< a, b, c, d, e, f >$, where $an^{\frac{2}{3}} \leq e < (a + 1)n^{\frac{2}{3}}$, and $bn^{\frac{2}{3}} \leq f < (b + 1)n^{\frac{2}{3}}$.

In the case $\varepsilon = \frac{2}{3}$ Lemma 5.6 yields

$$\begin{aligned} a' &= \left\lfloor \frac{z}{n^{\frac{1}{3}}} \right\rfloor \\ b' &= z \bmod n^{\frac{1}{3}} \\ c' &= \left\lfloor \frac{z'}{n^{\frac{1}{3}}} \right\rfloor \\ d' &= z' \bmod n^{\frac{1}{3}} \\ e' &= e + (c' - a)n^{\frac{2}{3}} \\ f' &= f + (d' - b)n^{\frac{2}{3}} \\ z &= (cn^{\frac{1}{3}} + d + e) \bmod n^{\frac{2}{3}} \\ z' &= (an^{\frac{1}{3}} + b + f) \bmod n^{\frac{2}{3}}. \end{aligned}$$

The mappings $e \mapsto (cn^{\frac{1}{3}} + d + e) \bmod n^{\frac{2}{3}}$ from $\{an^{\frac{2}{3}}, \ldots, (a+1)n^{\frac{2}{3}} - 1\}$ to $[n^{\frac{2}{3}}]$ and $f \mapsto (an^{\frac{1}{3}} + b + f) \bmod n^{\frac{2}{3}}$ from $\{bn^{\frac{2}{3}}, \ldots, (b+1)n^{\frac{2}{3}} - 1\}$ to $[n^{\frac{2}{3}}]$ are bijective. Hence

$$\begin{array}{ccc} \{an^{\frac{2}{3}}, \ldots, (a+1)n^{\frac{2}{3}} - 1\} \times \{bn^{\frac{2}{3}}, \ldots, (b+1)n^{\frac{2}{3}} - 1\} & \longrightarrow & [n^{\frac{1}{3}}]^4 \\ (e, f) & \mapsto & (a', b', c', d') \end{array}$$

is a bijective mapping. Thus for all $B, B' \in \mathcal{B}$

$$|ata(\{(B, BR) \mid BR \text{ is brick in } B\}) \cap \{(B', BR) \mid BR \text{ is brick in } B'\}| = 1$$

holds.

Now we show that *ata* is bijective. From the definition of algorithm **Brick Transport** follows that step 1 and step 4 define permutations on $\mathcal{B}$. As noted in the proof of Lemma 5.6 the combination of step 2 and step 3 defines a permutation on $\mathcal{B}$. So *ata* is bijective. $\diamond$

The above proof can be extended to the case where $n^{2(1-\varepsilon)}$ divides n^{ε}. In such a case every block receives $n^{6(\varepsilon - \frac{2}{3})}$ bricks from every block, i.e., *ata* is bijective and for all $B, B' \in \mathcal{B}$:

$$|ata(\{(B, BR) \mid BR \text{ is brick in } B\}) \cap \{(B', BR) \mid BR \text{ is brick in } B'\}| = n^{6(\varepsilon - \frac{2}{3})}.$$

To see this, note that for $e \in [n^\varepsilon]$ the mapping

$$e \mapsto (cn^{1-\varepsilon} + d + e - an^\varepsilon) \bmod n^{2(1-\varepsilon)}$$

hits every value in $[n^{2(1-\varepsilon)}]$ exactly $n^{3(\varepsilon-\frac{2}{3})}$ times.

In such a case we combine $n^{6(\varepsilon-\frac{2}{3})}$ bricks to one large brick. Note that the number of large bricks is equal to the number of blocks and that in the case $\varepsilon = \frac{2}{3}$ a large brick consists of one brick.

5.4.6 The Sorting Algorithm.

An indexing of the blocks in the four-dimensional mesh is also an indexing of the blocks in the OTIS-Mesh. We assume that g is a continuous indexing of the blocks in the four-dimensional mesh. The algorithm sorts the OTIS-Mesh with respect to the indexing induced by g, i.e., the index of a place $((a,b,c,d),l)$, $(a,b,c,d) \in [n]^4$, $l \in [k]$ is $k \cdot |B| \cdot g(B) + h_B((a,b,c,d),l)$. Here B is the block in which (a,b,c,d) lies, $g(B)$ is the index of block B, and h_B is an indexing of the places in B.

For a block B, we denote by ${}^g ata_B(j)$ the brick in B whose packets are transported to block j with respect to g if algorithm **Brick Transport** is performed.

We use blocks of side length $n^{\frac{2}{3}}$ and sort them with the simulation technique of Section 5.4.3. As all-to-all mapping we use **Brick Transport** for $\varepsilon = \frac{2}{3}$.

Algorithm Sort.

1. (a) Sort all blocks.
   (b) Within all blocks B: Transport the packet with the i-th largest key in the block to brick ${}^g ata_B(i \bmod n^{\frac{4}{3}})$, $0 \le i < kn^{\frac{8}{3}}$.
2. Perform an all-to-all mapping (**Brick Transport**, $\varepsilon = n^{\frac{2}{3}}$).

3. (a) Sort all blocks.

   (b) Within all blocks B do: Transport the packet with the i-th largest key in the block to brick ${}^{g}ata_B(\left\lfloor \frac{i}{kn^{\frac{4}{3}}} \right\rfloor)$, $0 \leq i < kn^{\frac{8}{3}}$.

4. Perform an all-to-all mapping (**Brick Transport**, $\varepsilon = n^{\frac{2}{3}}$).

5. Sort all pairs of blocks $(2i, 2i+1)$ (with respect to the indexing within the blocks), $0 \leq i < \frac{n^{\frac{4}{3}}}{2}$. Sort all pairs of blocks $(2i-1, 2i)$, $0 < i < \frac{n^{\frac{4}{3}}}{2}$ (with respect to the indexing within the blocks), (pairs of blocks with respect to g).

Theorem 5.8 *For all k algorithm* **Sort** *is a k-k sorting algorithm on the OTIS-Mesh with a buffer size of $k+4$. It has a running time of $8n + o(n)$ steps, for $k \leq 4$, and $2kn + o(kn)$ steps, for $k \geq 4$.*

Proof:
The OTIS-$M_{2,n}$ network consists of n^4 nodes. In the case $\varepsilon = \frac{2}{3}$ we have $n^{\frac{4}{3}}$ blocks and each block has $n^{\frac{4}{3}}$ bricks. For all k the inequality 5.1 is fulfilled. By [25] and the discussion in Section 5.4.2 algorithm **Sort** solves the k-k sorting problem.

Step 1a and step 3a are k-k sortings within blocks and steps 1b and step 3b are k-k routings within blocks. So these steps can be performed with the simulation technique of Section 5.4.3. Hence steps 1, 3, 5 can be achieved in $o(kn)$ steps with the buffer size $k+4$ (e.g. see [25, 19, 7]). For step 2 and step 4 see Theorem 5.5. ◇

The generalization of algorithm **Sort** to blocks with side length n^{ε}, $\frac{2}{3} \leq \varepsilon < 1$, where $n^{2(1-\varepsilon)}$ divides n^{ε} is straight forward and is left out.

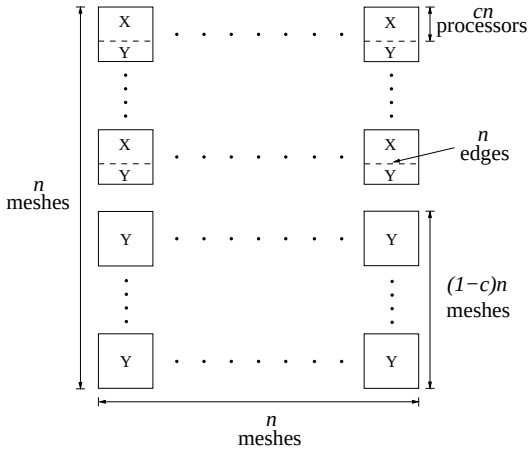

Figure 5.4: An OTIS-Mesh divided in two areas X and Y.

5.4.7 A Lower Bound for Sorting on the OTIS-Mesh.

Now we prove a lower bound for k-k routing and k-k sorting. The proof is based on a bisection argument. Note that in the following proof only electronical links connect the two areas. Hence this lower bound is independent of the bandwidth of the optical links.

Theorem 5.9 *Any k-k sorting (routing) algorithm on the OTIS-Mesh requires at least* $\max\{4n-3, \frac{1}{\sqrt{2}}\,kn\ (\approx 0.707kn)\}$ *steps in the worst case.*

Proof:

Look at Figure 5.4. There are two areas of processors denoted with X and Y. For $c = \frac{1}{\sqrt{2}}$ both areas consist of c^2n^4 processors. Note that there is no optical link between a processor of area X and processor of area Y. We have $C_{O_{M_{2,n}}}(X,Y) = cn^3$. Hence $bw(O_{M_{2,n}}) \leq \frac{1}{cn}$. Note that $4n-3$ is the diameter of OTIS-$M_{2,n}$ (see Theorem 5.18). $\diamond$

5.4.8 A Comparison with Mesh Algorithms.

In this section, we compare algorithm **Sort** with sorting algorithms designed for meshes. Although the OTIS-$M_{2,n}$ network consists of n^2 two-dimensional meshes $M_{2,n}$ it is not fair to compare **Sort** with sorting algorithms for two-dimensional meshes since we have an $\Omega(kn^2)$ lower bound for k-k sorting on M_{2,n^2}. A reasonable mesh for a comparison is a four-dimensional mesh. The structure of the OTIS-Mesh is similar to the structure of a four-dimensional mesh but there are also differences. $M_{4,n}$ and the OTIS-$M_{2,n}$ network have the same number of processors and the same diameter (up to one) but the OTIS-Mesh has a smaller bisection width (see Section 5.4.7) and its degree and number of links is by a factor of $\frac{5}{8}$ smaller. Hence it is not surprising that algorithms for $M_{4,n}$ have a better running time.

We compare the performance of algorithm **Sort** with the fastest k-k sorting algorithms for the four-dimensional mesh. In the case $k = 1$ the fastest known 1-1 sorting algorithm for $M_{4,n}$ with buffer size $O(1)$ requires $5n+o(n)$ steps [48]. This algorithm makes copies of the packets during the sorting. It is by a factor of $\frac{5}{8}$ faster than **Sort**. The fastest algorithm on $M_{4,n}$ with buffer size $O(1)$ that does not make copies needs $6n + o(n)$ steps [48]. Both algorithms have a similar structure to **Sort**. The main difference lies step 2 and step 4 of **Sort**. In step 2 **Sort** distribute the packets in the whole network. The $M_{4,n}$ algorithm in [48] distributes the packets within a *centre region* of the network and in step 4 it routes the packets from the centre region to their destination. As a consequence packets have to travel shorter distances and so the faster running times are achieved. The centre region consists of all processors that have a distance of at most n from the centre of $M_{4,n}$ (processor $(\lfloor \frac{n}{2} \rfloor, \lfloor \frac{n}{2} \rfloor, \lfloor \frac{n}{2} \rfloor, \lfloor \frac{n}{2} \rfloor)$). This region contains about half of the processors of $M_{4,n}$. Due to the structure of the OTIS-Mesh it seems hard to find such a region in it. Furthermore, step 2 and step 4 of the $M_{4,n}$

algorithm uses the fact that two so called *unshuffle permutations* can be routed distance-optimally on $M_{4,n}$. This requires the full link capacity of the mesh (i.e. all 8 directions). Hence it seems unlikely that this method can be used to design a faster algorithm for the OTIS-Mesh. Finally, an algorithm based on this method need a larger buffer size than **Sort** since we have to store more than one packet per processor in the centre region.

Sorting on $M_{4,n}$ in the case $k \in \{2, \ldots, 15\}$ has received less attention in the literature. Hence we do not compare our algorithm for this case.

For the case $k \geq 16$ the sorting algorithm for $M_{4,n}$ of Kunde in [25] asymptotically matches the bisection bound of $\frac{kn}{2}$. Hence it is four times faster than **Sort**. The algorithm has a buffer size of k and uses sorting with all-to-all mappings. The running time of **Sort** is a factor of at most $2\sqrt{2}$ away from the bisection bound. The running time of **Sort** could (possibly) be reduced to the half by *overlapping* shifts, performed in step 1 and step 4 of algorithm **Brick Transport**, in different meshes of a superblock. For the remaining factor of $\sqrt{2}$ we do not know whether it exists due to the weakness of the lower or upper bound. Here further research is necessary.

If we assume a bandwidth of at least four and a k-k sorting algorithm for the four-dimensional mesh with a running time of $\max\{4n + o(n), k\frac{n}{2} + o(kn)\}$ steps, then the simulation (Theorem 5.4) of the algorithm would result in a $\max\{12n + o(n), 3k\frac{n}{2} + o(kn)\}$ step algorithm on the OTIS-Mesh. For $k \geq 8$ such an algorithm would be faster than algorithm **Sort**.

5.5 Oblivious Routing on OTIS-G Networks.

In this section, we give a $(|V|, |V|)$-partitioning of size $(|V|, |V|, O(|V|))$ of an OTIS-G network, where $G = (V, E)$. We use the set V as index set for the source blocks, destination blocks, entry places, and exit places.

1. The groups of OTIS-G are the source and destination blocks, i.e., for all $v \in V$ we set $S_v \stackrel{\text{def}}{=} \{v\} \times V$ and $D_v \stackrel{\text{def}}{=} \{v\} \times V$.

2. For all $v, v' \in V$, we set $entry_v(v') \stackrel{\text{def}}{=} (v, v') \in D_v$.

3. For all $v \in V$, we set $T_v \stackrel{\text{def}}{=} (V_v, E_v, exit_v)$, where $V_v = (\{v\} \times V) \cup (V \times \{v\})$, $exit_v = (v, v) \in S_v$, and $E_v = E_{sp} \cup \{((v, v'), (v', v)) \mid v' \in V, v \neq v'\}$. Here E_{sp} are the edges of a directed spanning tree of G_v with root (v, v) (i.e. a directed tree that consists of all nodes of G_v and has root (v, v)).

Note that $entry_{v'}(v) \in T_v$, for all $v' \in V$. T_v is a directed tree for all $v \in V$. We get:

$$((S_v)_{v \in V}, (D_v)_{v \in V}, (entry_v)_{v \in V}, (T_v)_{v \in V})$$

is a $(|V|, |V|)$-partitioning of OTIS-G of size

$$(|V|, |V|, O(|V|)).$$

By Theorem 4.3 and Theorem 4.1 we get:

Theorem 5.10 *Algorithm* **KKOblivious**, *given in Section 4.4.1, is an oblivious k-k routing algorithm on OTIS-G with running time $O(k|V_G|)$ and buffer size $O(k)$. A lower bound for oblivious k-k routing in OTIS-G is $\Omega(\frac{k|V_G|}{deg(G)})$.*

Proof:
An OTIS-G network consists of $|V_G|^2$ nodes. The lower bound follows from Theorem 4.1. The upper bound follows from Theorem 4.3. ◇

Corollary 5.11 *If G is of fixed degree, then an oblivious k-k routing algorithm on OTIS-G with running time $O(k|V_G|)$ and buffer size $O(k)$ exists. The running time asymptotically matches the lower bound of Theorem 5.10.*

5.6 Diameter of OTIS-G Networks.

In this section, we give the diameter of an OTIS-G network. In [50] the diameter of an OTIS-Mesh and in [52] the diameter of an OTIS-Hypercube was determined. Both results have similar proofs. These proofs have inspired the result of this section. In [5] the diameter of an OTIS-G network and a proof idea is given.[4]

We begin with a notation for shortest paths. We use $x \longleftrightarrow_g^* y$ for a shortest path between nodes x and y in G_g. In general, more than one shortest path between two nodes exists. Hence $x \longleftrightarrow_g^* y$ describes a set of paths. We use $\longleftrightarrow^*$ to describe a path w, e.g., we write path w as $x \longleftrightarrow_g^* y \longleftrightarrow_o z$. In such a case, we mean a path $w = w_1 w_2$, where w_1 is a shortest path between x and y (in G_g) and w_2 is a path between y to z that uses one optical link. All such paths have the same length. Hence $|w|$ is used todenote their length. It is possible that no such path exists (e.g. see Theorem 5.17). In such a case $|w| = \infty$.

We say a path p *uses* k *optical links* if $k = |\{e \in E_{O_G} : p \text{ uses } e \text{ and } e \text{ is an optical link } \}|$.

We analyse shortest paths in OTIS-G networks. A path in an OTIS-G network uses zero or more optical links. The following two lemmata show that a shortest path uses at most two optical links.

Lemma 5.12 *Let $O_G = (V, E)$ be an OTIS-G network and $s = (g, p)$, $t = (g', p') \in V$, p, g, p', $g' \in V_G$. Let w be a path between s and t in O_G that uses $k > 2, k$ odd, optical links.*

If $g \neq g'$, then a path w' between s and s exists such that

- *w' uses one optical link,*

[4] The result of this section was achieved independently of [5].

- $|w'| \leq |w| - k + 1$, *and*
- $|w'| = d_G(p, g') + d_G(g, p') + 1$.

If $g = g'$, *then a path* w' *between* s *and* t *exists such that*

- w *uses no optical link,*
- $|w'| \leq |w| - k$, *and*
- $|w'| = d_G(p, p')$.

Proof:

In the following, we decompose w in $k + 1$ subpaths which use no optical link and k paths of length one which use optical links.

Let $k = 2l - 1$, $l > 1$. W.l.o.g. we write w as

$$\begin{array}{lll} (g_0, p_0) & \longleftrightarrow^*_{g_0} & (g_0, p_1) \\ & \longleftrightarrow_o & (p_1, g_0) \\ & \longleftrightarrow^*_{p_1} & (p_1, g_1) \\ & \longleftrightarrow_o & (g_1, p_1) \\ & \vdots & \vdots \\ & \longleftrightarrow^*_{g_{l-1}} & (g_{l-1}, p_l) \\ & \longleftrightarrow_o & (p_l, g_{l-1}) \\ & \longleftrightarrow^*_{p_l} & (p_l, g_l), \end{array}$$

where $p_0, \ldots, p_l, g_0, \ldots, g_l \in V_G$, $p_i \notin \{g_{i-1}, g_i\}$, $i \in \{1, \ldots, l-1\}$, $p_l \neq g_{l-1}$, and $g_0 = g$, $p_0 = p, p_l = g', g_l = p'$.

In w (sub)paths between nodes p_i and p_{i+1} in group g_i, $i \in [l-1]$, exist. By the triangle inequality and the fact that all groups are isomorphic to G, the sum of the length of these paths is greater or equal to the length of a shortest path between p_0 and p_l in group g_0.

In w also (sub)paths between g_i and g_{i+1} in group p_{i+1}, $i \in [l]$, exist. The length of these paths is greater or equal to the length of a shortest path from g_0 to g_l in group p_l.

If $g = g'$, then $p_l = g_0 = g$ and we define $w' = (g_0, p_0) \longleftrightarrow_{g_0}^{*} (p_l, g_l)$. Thus $|w'| = d_G(p, p') \leq |w| - k$.

Otherwise let w' be

$$\begin{aligned} (g_0, p_0) &\longleftrightarrow_{g_0}^{*} (g_0, p_l) \quad \text{(path in group } g_0\text{)} \\ &\longleftrightarrow_{o} (p_l, g_0) \\ &\longleftrightarrow_{p_l}^{*} (p_l, g_l). \quad \text{(path in group } p_l\text{)} \end{aligned}$$

Since $g \neq g'$ we have $g_0 \neq p_l$ and hence w' exists. Path w uses $k > 2$ optical links and path w' one. Hence we get $|w'| \leq |w| - k + 1$. It holds

$$|w'| = d_G(p_0, p_l) + d_G(g_0, g_l) + 1 = d_G(p, g') + d_G(g, p') + 1.$$

$\diamond$

Lemma 5.13 *Let $O_G = (V, E)$ be an OTIS-G network and $s = (g, p)$, $t = (g', p') \in V$, p, g, p', $g' \in V_G$. Let w be a path between s and t in O_G that uses $k > 2$, k even, optical links.*

If $g \neq g'$ and $p \notin \{g, g'\}$, then a path w' between s and t exists such that

- *w' uses two optical link,*
- *$|w'| \leq |w| - k + 2$, and*
- *$|w'| = d_G(p, p') + d_G(g, g') + 2$.*

If $g \neq g'$, $p = g$, and $p \neq g'$, then a path w' between s and t exists such that

- *w' uses one optical link,*
- *$|w'| \leq |w| - k + 1$, and*

- $|w'| = d_G(p, p') + d_G(g, g') + 1 = d_G(g, p') + d_G(p, g') + 1$.

If $g \neq g'$, $p \neq g$, and $p = g'$, then a path w' between s and t exists such that

- *w' uses one optical link,*
- *$|w'| \leq |w| - k + 1$, and*
- $|w'| = d_G(p, g') + d_G(g, p') + 1 = d_G(g, p') + 1$.

If $g = g'$, then a path w' between s and t exists such that

- *w uses no optical links,*
- *$|w'| \leq |w| - k$, and*
- $|w'| = d_G(p, p')$.

Proof:
The proof can be done analogously to the proof of Lemma 5.12. Let $k = 2l$, $l > 1$. W.l.o.g. we write w as

$$\begin{array}{lll} (g_0, p_0) & \longleftrightarrow^*_{g_0} & (g_0, p_1) \\ & \longleftrightarrow_o & (p_1, g_0) \\ & \longleftrightarrow^*_{p_1} & (p_1, g_1) \\ & \longleftrightarrow_o & (g_1, p_1) \\ & \vdots & \vdots \\ & \longleftrightarrow^*_{p_l} & (p_l, g_l) \\ & \longleftrightarrow_o & (g_l, p_l) \\ & \longleftrightarrow^*_{g_l} & (g_l, p_{l+1}), \end{array}$$

where $p_0, \ldots, p_{l+1}, g_0, \ldots, g_l \in V_G$, $p_i \notin \{g_{i-1}, g_i\}$, $i \in \{1, \ldots, l\}$, and $g_0 = g$, $p_0 = p, g_l = g', p_{l+1} = p'$.

In w (sub)paths between p_i and p_{i+1} in group g_i, $i \in [l]$, exist. By the triangle inequality and the fact that all groups are isomorphic to G, the sum of the length of these paths is greater or equal to the length of a shortest path from p_0 to p_{l+1} in group p_0.

In w (sub)paths between g_i and g_{i+1} in group p_{i+1}, $i \in [l]$, exist. By the triangle inequality the sum of the length of these paths is greater or equal to the length of a shortest path from g_0 to g_l in group g_l.

If $p \notin \{g, g'\}$, then $p_0 \neq g_0$ and $p_0 \neq g_l$. Let w' be

$$\begin{aligned}(g_0, p_0) &\longleftrightarrow_o (p_0, g_0)\\ &\longleftrightarrow^*_{p_0} (p_0, g_l)\\ &\longleftrightarrow_o (g_l, p_0)\\ &\longleftrightarrow^*_{g_l} (g_l, p_{l+1})\end{aligned}$$

Path w uses $k > 3$ optical links and path w' two. Hence we get $|w'| \leq |w| - k + 2$. It holds

$$|w'| = d_G(g_0, g_l) + d_G(p_0, p_{l+1}) + 2 = d_G(g, g') + d_G(p, p') + 2.$$

If $p = g'$ and $p \neq g$, then $p_0 = g_l$ and $p_0 \neq g_0$. Let w' be

$$\begin{aligned}(g_0, p_0) &\longleftrightarrow_o (p_0, g_0)\\ &\longleftrightarrow^*_{p_0} (g_l, p_{l+1}).\end{aligned}$$

Path w' consists an optical link and a shortest path in g_0 between g_0 and p_{l+1}. This path is of shorter or equal length than a path in group g_0 from node g_0 to node $g_l = p_0$ to node p_{l+1}. Thus $|w'| \leq |w| - k + 1$ and $|w'| = d_G(p, g') + d_G(g, p') + 1 = d_G(g, p') + 1$.

If $p \neq g'$ and $p = g$, then $g_0 \neq g_l$ and $p_0 = g_0$. Let w' be

$$\begin{aligned}(g_0, p_0) &\longleftrightarrow^*_{g_0} (g_0, g_l)\\ &\longleftrightarrow_o (g_l, g_0)\\ &\longleftrightarrow^*_{g_l} (g_l, p_{l+1})\end{aligned}$$

Path w' consists of a shortest path in group g_0 between $p_0 = g_0$ and g_l, a shortest path in group g_l between $g_0 = p_0$ and p_{l+1} and one optical link. Thus $|w'| \leq |w| - k + 1$ and $|w'| = d_G(g, g') + d_G(p, p') + 1 = d_G(p, g') + d_G(g, p') + 1$.

If $p = g' = g$, then $w' = (g, p) \longleftrightarrow_g^* (g, p')$, $|w'| \leq |w| - k$, and $|w'| = d_G(p, p')$. $\diamond$

Now we determine a lower bound for the length of a path that uses one or two optical links. Remember that the only possibility to come from a group g to a group g', $g \neq g'$, is to use optical links.

Lemma 5.14 *Let $O_G = (V, E)$ be an OTIS-G network and $s = (g, p)$, $t = (g', p') \in V$, p, g, p', $g' \in V_G$. Let w be a path between s and t in O_G that uses one optical link. Then $g \neq g'$ and $|w| \geq d_G(g, p') + d_G(g', p) + 1$.*

Proof:

Path w uses one optical link. Assume $\{(g, x), (x, g)\}$, $x \in V_G$, is the optical link. Then $g \neq x$ by the definition of OTIS networks. Path w is a path between (g, p) and (g', p'). Hence $x = g'$. This yields $g \neq g'$.

Any path between s and t that uses one optical link has to use this link between nodes (g, g') and (g', g). Hence $|w| \geq d_G(p, g') + d_G(p', g) + 1$. $\diamond$

Lemma 5.15 *Let $O_G = (V, E)$ be an OTIS-G network and $s = (g, p)$, $t = (g', p') \in V$, p, g, p', $g' \in V_G$. Let w be a path between s and t in O_G that uses two optical links. Then $|w| \geq d_G(g, g') + d_G(p', p) + 2$.*

Proof:

Path w uses two optical links and is a path between (g, p) and (g', p'). Hence w uses one optical link between nodes (g, x) and (x, g) and another between the nodes (y, g') and (g', y), where $x \neq g$ and $y \neq g'$. Thus $x = y$ and $|w| \geq d_G(p, x) + d_G(g, g') + d_G(y, p') + 2 \geq d_G(p, p') + d_G(g, g') + 2$. $\diamond$

Now we consider the case where both nodes are in the same group. We show that the shortest path in such a case is a path within the group.

Lemma 5.16 *Let $O_G = (V, E)$ be an OTIS-G network and $s = (g, p)$, $t = (g, p') \in V$, p, g, $p' \in V_G$. A shortest path between s and t in O_G uses no optical link. Furthermore, $d_{O_G}(s, t) = d_G(p, p')$.*

Proof:
By Lemma 5.13 and Lemma 5.12 we know that a shortest path uses zero, one, or two optical links. By Lemma 5.14 we know that one link is not possible, by Lemma 5.15 we know that a path between s and t that uses two optical links has at least a length of $d_G(p, p') + 2$. In G_g a shortest path between s and t has length $d_G(p, p')$. ◇

The following theorem describes shortest paths in an OTIS-G network.

Theorem 5.17 (Shortest Paths in OTIS-G Networks.) *Let O_G be an OTIS-G network and $s = (g, p)$, $t = (g', p')$, p, g, p', $g' \in V_G$. Let path $p(s, t)$ be given as follows:*

If $g = g'$, then $p(s, t)$ is a shortest path between p and p' in G_g.

If $g \neq g'$, then $p(s, t)$ is the shortest of the two paths $p_1(s, t)$ and $p_2(s, t)$.

- $p_1(s,t) \stackrel{\text{def}}{=} (g,p) \longleftrightarrow_g^* (g,g') \longleftrightarrow_o (g',g) \longleftrightarrow_{g'}^* (g',p')$,
- $p_2(s,t) \stackrel{\text{def}}{=} (g,p) \longleftrightarrow_g^* (g,p') \longleftrightarrow_o (p',g) \longleftrightarrow_{p'}^* (p',g') \longleftrightarrow_o (g',p')$.

(In the case $p' \in \{g, g'\}$ the path $p_2(s, t)$ does not exist and we set $|p_2(s, t)| = \infty$.)

Then $|p(s, t)| = d_{O_G}(s, t)$.

Furthermore, if $g = g'$, then $p(s, t) = d_G(p, p')$ and if $g \neq g'$, then $|p(s, t)| = \min\{d_G(g, p') + d_G(g', p) + 1, d_G(p, p') + d_G(g, g') + 2\}$.

Proof:

The case $g = g'$ follows from Lemma 5.16.

In the case $g \neq g'$ a shortest path between s and t uses one or two optical links. Path $p_1(s,t)$ uses one optical link, is a path from s to t, and has length $d_G(p,g') + d_G(p',g) + 1$. By Lemma 5.14 there is no shorter path between s and t that uses one optical link.

If $p \notin \{g, g'\}$, then $p_2(s,t)$ exists, uses two optical links, is a path between s and t, and has length $d_G(g,g')+d_G(p,p')+2$. By Lemma 5.15 no shorter path between s and t that uses two optical links exists. If $p \in \{g, g'\}$, then $p_2(s,t)$ does not exist. In this case $d_G(g,g') + d_g(p,p') + 2 \geq d_G(p,g') + d_G(g,p') + 2$ and since $g \neq g'$ the shortest path between s and t uses one optical link and has length $d_G(p,g') + d_G(g,p') + 1$. ◇

Now we are able to determine the diameter of an OTIS-G network.

Theorem 5.18 (Diameter of OTIS-G ([5, 50, 52]).) *The diameter of an OTIS-G network is* $2D(G) + 1$.

Proof:

By Theorem 5.17 $D(O_G) \leq 2D(G) + 1$.

Remember that $|V_G| > 1$. Hence there exists nodes p, g, $p \neq g$ in V_G such that $d_G(p,g) = D(G)$. Consider nodes $s = (g,g)$ and $t = (p,p)$ in O_G. Theorem 5.17 yields $d_{O_G}(s,t) = 2d_G(p,g) + 1 = 2D(G) + 1$. ◇

The proof of the theorem shows that nodes (g,g) and (p,p) in an OTIS-G network have maximal distance when $d_G(g,p) = D(G)$. Furthermore, a node (g,g) is not incident to an optical link and has degree $deg_G(g) < deg(O_G)$. Any node (g,p), where $g \neq p$, is incident to one optical link and has degree $deg_G(p) + 1 \leq deg(O_G)$. This observation is the motivation for the introduction of Extended OTIS networks.

5.7 Extended OTIS-G Networks.

In this section, we reduce the diameter of OTIS-G networks. We extend the definition of an OTIS-G network by adding optical links to nodes (g, p), where $g = p$. These additional links do not increase the degree of the network.

5.7.1 Basic Definitions and Properties.

We begin with the definition of an Extended OTIS-G network.

Definition 5.19 (Extended OTIS-G network using f.)
Let $G = (V_G, E_G)$, $|V_G| > 1$, be a graph and $f : V_G \longrightarrow V_G$ be a mapping such that $f \circ f = id_{V_G}$. The Extended OTIS-G network using f (or $X_{G,f}$ network for short) is a triple (V, E, f), where (V, E) is an undirected graph with node set $V = V_G \times V_G$ and edge set $E = E_e \cup E_o \cup E_f$, where

$$\begin{aligned} E_e &= \{\{(g,p),(g,p')\} \mid g,p,p' \in V_G \wedge \{p,p'\} \in E_G\}, \\ E_o &= \{\{(g,p),(p,g)\} \mid p,g \in V_G,\ p \neq g\}, \text{ and} \\ E_f &= \{\{(g,g),(f(g),f(g))\} \mid g \in V_G, f(g) \neq g\}. \end{aligned}$$

We call edges in E_o optical links, edges in E_e electronical links, and edges in E_f f-links. We use $\longleftrightarrow_o$ to denote optical links, $\longleftrightarrow_e$ to denote electronical links, and $\longleftrightarrow_f$ to denote f-links.

We observe some simple properties following directly from the definition of an $X_{G,f}$ network (V, E, f):

- $|V| = |V_G|^2 = |V_{O_G}|$.
- $|E| = |V_G||E_G| + \frac{|V_G|^2 - |V_G|}{2} + \frac{|\{x|f(x)\neq x\}|}{2} = |E_{O_G}| + \frac{|\{x|f(x)\neq x\}|}{2}$.
- $deg_{X_{G,f}}((g,p)) = deg_{O_G}((g,p)) = deg_G(p) + 1$ if $g \neq p$.

- $deg_{X_{G,f}}((g,g)) = deg_{O_G}((g,g)) + 1 = deg_G(g) + 1$ if $f(g) \neq g$.
- $deg_{X_{G,f}}((g,g)) = deg_{O_G}((g,g)) = deg_G(g)$ if $f(g) = g$.
- If f has no fixed-point and G is a regular graph, then $X_{G,f}$ is a $deg(G)+1$-regular graph.
- $X_{G,id}$ is isomorphic to O_G.
- $D(X_{G,f}) \leq 2D(G) + 1$, [5, 50], Theorem 5.18.
- An OTIS-G network can be embedded into an Extended OTIS-G network using f with dilation, congestion and load one. Hence algorithms for OTIS-G networks can be performed in Extended OTIS-G networks using f without any delay.

We assume $f \neq id$. Thus an $x \in V_G$ exists such that $f(x) \neq x$. In the case $f = id$ is allowed, we note it explicitly.

5.7.2 Shortest Paths in Extended OTIS-G Networks.

Before we start, we introduce some notations for paths. We use $x \longleftrightarrow^* y$ for a shortest path between x and y in $X_{G,f}$, $x \longleftrightarrow^*_{-f} y$ for a shortest path between x and y in $X_{G,f}$ taken from the set of all paths between x and y in $X_{G,f}$ without an f-link, and $x \longleftrightarrow^*_g y$ for a shortest path between x and y in G_g.

We say a path p *uses* k *f-links* if $k = |\{e \in E_{X_{G,f}} : p \text{ uses } e \text{ and } e \text{ is an } f\text{-link }\}|$ edges of the path are f-links.

A path in $X_{G,f}$ that uses no f-link is a path in O_G, and a path in O_G is a path in $X_{G,f}$ that uses no f-link. We get the following lemma.

Lemma 5.20 *Let (V, E, f) be an Extended OTIS-G network using f, $s = (g, p)$, $t = (g', p') \in V$, p, g, p', $g' \in V_G$ and $p(s, t)$ as in Theorem 5.17.*

Then $|p(s,t)| = d_{O_G}(s,t) = |s \longleftrightarrow^*_{-f} t|$.

Proof:
Path $p(s,t)$ uses no f-link. Hence $|p(s,t)| \geq |s \longleftrightarrow^*_{-f} t|$. A path between s and t in $X_{G,f}$ that uses no f-link is a path between s and t in O_G. Hence $|s \longleftrightarrow^*_{-f} t| \geq d_{O_G}(s,t)$. Theorem 5.17 yields $d_{O_G}(s,t) = |p(s,t)|$. $\diamond$

A path that uses k f-links can be decomposed into $k+1$ subpaths that use no f-link and k paths of length one that use one f-link. These $k+1$ subpaths have a special form. Their first or last node or both of them have the form (x,x), $x \in V_G$. Before we begin to analyse the length of paths with f-links, we analyse these special paths. The following lemma can be obtained by Theorem 5.17.

Lemma 5.21 *Let* (V,E,f) *be an Extended OTIS-G network using* f *and* $s = (g,p)$, $t = (g',p') \in V$, $p, g, p', g', g \neq g' \in V_G$. *Then the following holds:*

- *If* $p = g$ *or* $p' = g'$, *then* $d_{O_G}(s,t) = d_G(p,p') + d_G(g,g') + 1$.
- *If* $p = g$ *and* $p' = g'$, *then* $d_{O_G}(s,t) = 2d_G(p,p') + 1 = 2d_G(g,g') + 1 = 2d_G(p,g') + 1 = 2d_G(p',g) + 1$.

In both cases a shortest path between s *and* t *uses one optical link.*

Now we analyse paths using exactly one f-link.

Lemma 5.22 *Let* (V,E,f) *be an Extended OTIS-G network using* f, $s = (g,p)$, $t = (g',p')$, $X = (x,x) \in V, p, g, p', g', x \in V_G$, $f(x) \neq x$. *Let* $p(s,t,X)$ *be a path of the following form.*

$$p(s,t,X) \stackrel{\text{def}}{=} s \longleftrightarrow^*_{-f} (x,x) \longleftrightarrow_f (f(x),f(x)) \longleftrightarrow^*_{-f} t$$

Then path $p(s,t,X)$ is a path between s and t. It uses one f-link (at node X) and has length $d_G(g,x) + d_G(p,x) + d_G(g', f(x)) + d_G(p', f(x)) + 1 + \delta_x$, where

$$\delta_x = \begin{cases} 0 & : \quad x = g \quad \text{and} \quad f(x) = g' \\ 1 & : \quad x = g \quad \text{xor} \quad f(x) = g' \\ 2 & : \quad x \neq g \quad \text{and} \quad f(x) \neq g'. \end{cases}$$

Proof:
By Lemma 5.20 and Lemma 5.21:

$$\begin{aligned} |p(s,t,X)| &= d_{O_G}((g,p),(x,x)) + d_{O_G}((f(x),f(x)),(g',p')) + 1 \\ &= d_G(g,x) + d_G(p,x) + d_G(f(x),g') + d_G(f(x),p') + 1 + \delta_x. \end{aligned}$$

◇

It is obvious that the number of used f-links used by a shortest path in an Extended OTIS-G network depends on the choice of f. A desirable property of a shortest path would be if it uses at most one f-link. This can be achieved if we choose f such that it respects d_G.

Definition 5.23 *Let $G = (V_G, E_G)$ be a graph and $f : V_G \longrightarrow V_G$ a mapping. We say f respects d_G if and only if $d_G(x,y) = d_G(f(x), f(y))$ for all $x, y \in V_G$.*

We observe a simple property.

Lemma 5.24 *Let $G = (V_G, E_G)$ be a graph and $f \circ f = id_{V_G}$. Then f respects $d_G \iff d_G(x, f(y)) = d_G(f(x), y)$ for all $x, y \in V_G$.*

Proof:
Let $x, y \in V_G$ and $a = f(x)$, $b = f(y)$.

$\Longrightarrow$: $d_G(x, f(y)) = d_G(f(x), f(f(y))) = d_G(f(x), y)$.

$\Longleftarrow$: $d_G(x, y) = d_G(x, f(b)) = d_G(f(x), b) = d_G(f(x), f(y))$. $\diamond$

Now we prove that a shortest path in $X_{G,f}$ uses at most one f-link provided f respects d_G.

Lemma 5.25 *Let (V, E, f) be an $X_{G,f}$ network, f respects d_G, and $s = (g, p)$, $t = (g', p') \in V$, g, p, g', $p' \in V_G$.*

If a path w between s and t in $X_{G,f}$ exists that uses $k \geq 2$ f-links, then there exists a path w' between s and t such that w' uses at most one f-link and $|w'| < |w|$.

Proof:

W.l.o.g. we can assume that w is of the following form

$$\begin{array}{lllll} (g,p) & \longleftrightarrow_{-f}^{*} & (m_1,m_1) & \longleftrightarrow_{f} & (f(m_1),f(m_1)) \\ & \longleftrightarrow_{-f}^{*} & (m_2,m_2) & \longleftrightarrow_{f} & (f(m_2),f(m_2)) \\ & \longleftrightarrow_{-f}^{*} & \vdots & \vdots & \vdots \\ & \longleftrightarrow_{-f}^{*} & (m_k,m_k) & \longleftrightarrow_{f} & (f(m_k),f(m_k)) \\ & \longleftrightarrow_{-f}^{*} & (g',p'), \text{ where} & & \end{array}$$

$m_1, \ldots, m_k \in V_G$, and $\{m_1, \ldots, m_k\} \cap \{f(m_1), \ldots, f(m_k)\} = \emptyset$.

Lemma 5.20 provides

$$\begin{array}{rcll} |w| & = & d_{O_G}((g,p),(m_1,m_1)) & + \\ & & \sum_{i=1}^{k-1} d_{O_G}((f(m_i),f(m_i)),(m_{i+1},m_{i+1})) & + \\ & & k & + \\ & & d_{O_G}((f(m_k),f(m_k)),(g',p')). & \end{array}$$

By Lemma 5.21 we get

$$\begin{array}{rcl} d_{O_G}((g,p),(m_1,m_1)) & \geq & d_G(p,m_1), \\ d_{O_G}((f(m_{i+1}),f(m_{i+1})),(m_i,m_i)) & = & 2d_G(m_i,f(m_{i+1}))+1, \text{ and} \\ d_{O_G}((f(m_k),f(m_k)),(g',p')) & \geq & d_G(f(m_k),p'). \end{array}$$

Thus

$$\begin{array}{rcll} |w| & \geq & d_G(p, m_1) & + \\ & & 2\sum_{i=1}^{k-1} d_G(m_i, f(m_{i+1})) & + \\ & & 2k-1 & + \\ & & d_G(p', f(m_k)). & \end{array}$$

For even $k \geq 2$, we construct a path w' between s and t. Path w' uses no f-link and is of equal or shorter length than w. For odd $k \geq 3$, we decompose w into a path p_{k-1} using $k-1 \geq 2$ f-links and a path using one f-link. Then we apply the result for even k on p_{k-1}. So we can assume that $k \geq 2$ is even. We first consider the case $g \neq g'$:

$$\begin{array}{rcll} w' & = & (g,p) & \longleftrightarrow_g^* \quad (g, m_1) \\ & & & \longleftrightarrow_g^* \quad (g, f(m_2)) \\ & & & \longleftrightarrow_g^* \quad (g, m_3) \\ & & & \quad \vdots \qquad\quad \vdots \\ & & & \longleftrightarrow_g^* \quad (g, m_{k-1}) \\ & & & \longleftrightarrow_g^* \quad (g, f(m_k)) \\ & & & \longleftrightarrow_g^* \quad (g, g') \\ & & & \longleftrightarrow_o \quad (g', g) \\ & & & \longleftrightarrow_{g'}^* \quad (g', m_1) \\ & & & \longleftrightarrow_{g'}^* \quad (g', f(m_2)) \\ & & & \longleftrightarrow_{g'}^* \quad (g', m_3) \\ & & & \quad \vdots \qquad\quad \vdots \\ & & & \longleftrightarrow_{g'}^* \quad (g', m_{k-1}) \\ & & & \longleftrightarrow_{g'}^* \quad (g', f(m_k)) \\ & & & \longleftrightarrow_{g'}^* \quad (g', p'). \end{array}$$

We get for the length of w':

$$
\begin{aligned}
|w'| \quad = \quad & d_G(p, m_1) & + \\
& \sum_{\substack{i=1 \\ i \text{ odd}}}^{k-1} d_G(m_i, f(m_{i+1})) & + \\
& \sum_{\substack{i=2 \\ i \text{ even}}}^{k-2} d_G(f(m_i), m_{i+1}) & + \\
& 1 & + \\
& \sum_{\substack{i=1 \\ i \text{ odd}}}^{k-1} d_G(m_i, f(m_{i+1})) & + \\
& \sum_{\substack{i=2 \\ i \text{ even}}}^{k-2} d_G(f(m_i), m_{i+1}) & + \\
& d_G(p', f(m_k)).
\end{aligned}
$$

Function f respects d_G. We obtain $d_G(m_i, f(m_{i+1})) = d_G(f(m_i), m_{i+1})$. So $|w'| < |w|$.

In the case $g = g'$ the path w' is the nearly the same as in the case $g \neq g'$. Only the optical link between (g, g') and (g', g) is not used. So we have $|w'| < |w|$ for this case. ◇

Plugging the results of the above lemmata together we get:

Theorem 5.26 (Shortest Paths in Extended OTIS-G Networks.) *Let $G = (V_G, E_G)$ be a connected graph, $X_{G,f} = (V, E, f)$, f respects d_G, and $s = (g, p)$, $t = (g', p') \in V$, g, p, g', $p' \in V_G$. If $g \neq g'$, then a shortest path between s and t has length* $\min\{1 + d_G(g', p) + d_G(g, p'),\ 2 + d_G(p, p') + d_G(g, g'),\ 1 + \min\{d_G(g, x) + d_G(p, x) + d_G(g', f(x)) + d_G(p', f(x)) + \delta_x \mid x \in V_G, f(x) \neq x\}\}$. *If $g = g'$, then a shortest path between s and t has length $d_G(p, p')$.*

Proof:

Due to Lemma 5.25, we have to consider paths using at most one f-link.

Case $g \neq g'$: By Lemma 5.22 a shortest path between s and t using one f-link at node (x, x) has length $1 + d_G(g, x) + d_G(p, x) + d_G(g', f(x)) +$

$d_G(p', f(x)) + \delta_x$. Lemma 5.20 provides that

$$|s \longleftrightarrow^*_{-f} t| = 1 + \min\{d_G(g', p) + d_G(g, p'), 2 + d_G(p, p') + d_G(g, g')\}.$$

Case $g = g'$: By Lemma 5.22 a path between s and t that uses one f-link has a length $> d_G(p, p')$. $\diamond$

5.7.3 Diameter of Extended OTIS-G Networks.

Theorems 5.26 and 5.17 give information about the length of a shortest path between two nodes. This will be used in the following sections to give upper and lower bounds for the diameter of $X_{G,f}$ for some graphs G. A trivial upper bound for the diameter of $X_{G,f}$ is $2D(G) + 1$, (Theorem 5.18).

We begin with a lower bound for the diameter of $X_{G,f}$ ($f = id$ included). In the following $\mathcal{F}_G = \{f \mid f : V_G \longrightarrow V_G,\ f \circ f = id\}$.

Theorem 5.27 *For all $f \in \mathcal{F}_G$: $D(X_{G,f}) \geq D(G) + 1$.*

Proof:

Case $|V_G| = 2$: There are two possibilities for $X_{G,f}$. See Figure 5.5, (b), (c). In (c) we have $f(0) = 1$ and $f(1) = 0$. In (b) and (c) the diameter is at least $D(G) + 1$.

Case $|V_G| > 2$: Choose $g, p \in V_G$ such that $d_G(p, g) = D(G)$. There exists a $p' \in V_G - \{p, g\}$ such that $\{p, p'\} \in E_G$ or $\{g, p'\} \in E_G$. W.l.o.g. we assume that $\{p, p'\} \in E_G$. So $d_G(g, p') \geq D(G) - 1$. Set $s = (g, p)$ and $t = (p', p)$.

We have $g \neq p'$. Hence a shortest path w between s and t in X uses f-links or optical links. If it uses optical links but no f-links, then $d_{X_{G,f}}(s, t) = \min\{1 + d_G(g, p) + d_G(p', p), 2 + d_G(g, p') + d_G(p, p)\} \geq D(G) + 1$. Now we consider the case that w uses at least one f-link. Path w is undirected. Hence it describes a path from s to t and a path from t to s. Consider for

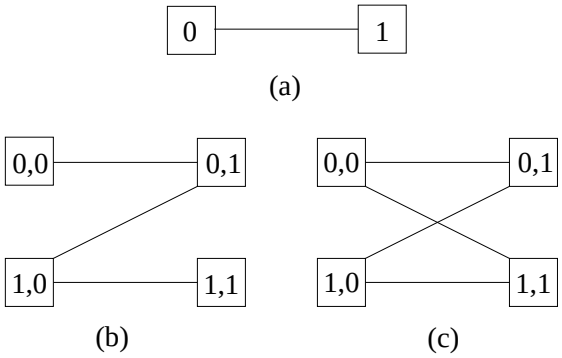

Figure 5.5: Extended OTIS-G networks, $|V_G| = 2$.
Case $|V_G| = 2$ in proof of Theorem 5.27.
(a) G, (b) $X_{G,id}$, (c) $X_{G,f}$.

a moment the (directed) path from s to t. Let (x, x) be the node where the first f-link starts and let (y, y) be the node where the last f-link ends. Further, let w_1 be the (sub)path (in w) between s and (x, x) and w_2 be the (sub)path (in w) between (y, y) and t. Path w is a shortest path and hence also w_1 and w_2 are shortest paths. We have $|w| \geq |w_1| + |w_2| + 1$. In w_1 and w_2 no f-link is used. From Lemma 5.21 follows that $|w_1| \geq d_G(p, g) = D(G)$ and $|w_2| \geq d_G(p, p') = 1$. Therefore $|w| \geq D(G) + 2$. ◇

For the example in Figure 5.5 $D(X_{G,f}) = D(G) + 1$ holds.

In the following, we analyse the diameter of $X_{G,f}$ for some special graphs G. We consider hypercubes, rings, and meshes. We give f such that the diameter of $X_{G,f}$ is optimal, i.e. for all $f' \in \mathcal{F}_G$ we have $D(X_{G,f}) \leq D(X_{G,f'})$.

5.7.3.1 Hypercubes.

Definition 5.28 (Hypercube) *An n-dimensional hypercube H_n=(V_n, E_n) is a graph with node set $V_n = \{0, 1\}^n$ and edge set*

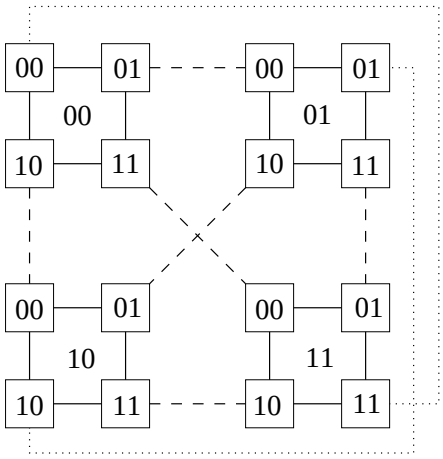

Figure 5.6: HCN(2,2).

This figure shows optical links as dashed lines, f-links as dotted lines, and electronic links as solid lines.

$$E_n = \{\{(x_0, \ldots x_{n-1}), (y_0, \ldots, y_{n-1})\} \mid x_i, y_j \in \{0,1\}, \textstyle\sum_{k\in[n]} |x_k - y_k| = 1\}.$$

The distance between nodes $x, y \in \{0,1\}^n$ in H_n is the number of different bits of x and y, i.e. $d_{H_n}(x,y) = \sum_{k\in[n]} |x_i \text{ xor } y_i|$. Hence the diameter of H_n is n.

Let $\overline{x}$ be the complement of $x \in \{0,1\}$ and $\overline{x} = (\overline{x}_0, \ldots, \overline{x}_{n-1})$, $x \in \{0,1\}^n$. For $x \in \{0,1\}^n$, we set $f_n(x) \stackrel{\text{def}}{=} \overline{x}$. Note that f_n respects d_{H_n}. The Extended OTIS-H_n network using f_n is isomorphic to the Hierarchical Cubic Network of size n denoted by $HCN(n,n)$ in [57]. In [57] it was shown that the diameter of $HCN(n,n)$ is $n+\lfloor(n+1)/3\rfloor+1$. Figure 5.6 shows an $HCN(2,2)$ network.

Theorem 5.29 *For all $f \in \mathcal{F}_{H_n}$: $D(X_{H_n,f}) \geq n + \lfloor(n+1)/3\rfloor + 1$.*

Proof:

The proof uses ideas of the proof of Theorem 5.27.

Let s, t be two nodes of X_{H_n}, $s = (g, p)$, $t = (g', p')$, w a shortest path between s and t, and $m = \lfloor (n+1)/3 \rfloor$. First, we assume that $n > 1$ and hence $m > 0$.

Case $n \bmod 3 = 0$: In this case $4m+1 = n + \lfloor (n+1)/3 \rfloor + 1$. Set $g = 0^{3m}$, $p = 0^m 1^m 1^m$, $g' = 1^m 1^m 0^m$ and $p' = 1^m 0^m 1^m$. If there are no f-links in w, then $|w| = \min\{1+d_{H_n}(g,p')+d_{H_n}(g',p), 2+d_{H_n}(g,g')+d_{H_n}(p,p')\} = 4m+1$. If there are f-links in w, we choose w_1, w_2 and x, y as in the proof of theorem 5.27. We get $|w| \geq |w_1| + |w_2| + 1 = d_{H_n}(g,p) + d_{H_n}(p,p') + 1 = 4m+1$.

Case $n \bmod 3 = 1$: In this case $4m+2 = n + \lfloor (n+1)/3 \rfloor + 1$. Set $g = 0^{3m+1}$, $p = 0^m 1^m 1^{m+1}$, $g' = 1^m 1^m 0^{m+1}$ and $p' = 1^m 0^m 1^{m+1}$. We get $|w| \geq 4m+2$.

Case $n \bmod 3 = 2$: In this case $4m = n + \lfloor (n+1)/3 \rfloor + 1$. This case is a little bit more complicated. To show the bound we have to consider two pairs of source and destination processors. First, set $g = 0^{3m-1}$, $p = 0^m 1^m 1^{m-1}$, $g' = 1^m 1^m 0^{m-1}$ and $p' = 1^m 0^m 1^{m-1}$. If w uses no f-link, then $|w| \geq 4m$. If w uses an f-link, then $|w| \geq 4m$, provided $f(g) \neq g'$. If $f(g) = g'$, then we can only conclude that $|w| \geq 4m-1$. Now, we consider $s_1 = (g_1, p_1)$, $t_1 = (g'_1, p'_1)$, $g_1 = g$, $p_1 = 1^{m-1} 0^m 1^m$, $g'_1 = 1^{m-1} 1^m 0^m$, $p'_1 = 0^{m-1} 1^m 1^m$. Again we are able to conclude that a shortest path between s_1 and t_1 that uses no f-link has length $\geq 4m$ and that a shortest path between s_1 and t_1 that uses an f-link has length $\geq 4m$, provided $f(g) \neq g'_1$. One of the distances $d_{X_{H_n,f}}(s,t)$ or $d_{X_{H_n,f}}(s_1,t_1)$ is at least $4m$.

If $n = 1$, then $n + \lfloor (n+1)/3 \rfloor + 1 = 2$. All possibilities for $X_{H_1,f}$ can be seen in Figure 5.5. $\diamond$

Corollary 5.30 *The function f_n is optimal for the diameter of the Extended OTIS-H_n network using f, i.e., for all $f \in \mathcal{F}_{H_n}$ we have $D(X_{H_n,f}) \geq D(X_{H_n,f_n})$.*

Corollary 5.31 *Any addition of links to an OTIS-H_n network that does not increase the degree of the network results in a diameter of at least* $n + \lfloor (n+1)/3 \rfloor + 1$.

Proof:
The n-dimensional hypercube is n-regular. Nodes (g, p), g, $p \in V_n$, $p \neq g$, of the OTIS-H_n network have degree $n+1$. Nodes (g, g), $g \in V_n$, of the OTIS-H_n network have degree n. Therefore, without increasing the degree of the network, links can only added between nodes in $\{(g, g) \mid g \in V_n\}$. $\diamond$

5.7.3.2 Rings.

Definition 5.32 (Ring) *A ring $R_n = (V_n, E_n)$ of size $n > 1$ is a graph with node set $V_n = [n]$ and edge set $E_n = \{\{i, (i+1) \bmod n\} \mid i \in [n]\}$.*

In a ring of size n the length of a shortest path between node x and node y is $\min\{|x-y|, n-|x-y|\}$. If n is an odd integer, then the diameter of R_n is $\frac{n-1}{2}$. If n is even, then $D(R_n) = \frac{n}{2}$.

Figure 5.7 shows an Extended OTIS-R_3 network using f, where $f(0) = 2$, $f(1) = 1$, and $f(2) = 0$.

In a ring of size n, for all $x \in V_n$ an $y \in V_n$ exists such that $d_{R_n}(x, y) = D(R_n)$. Hence in an OTIS-R_n network for each node (x, x) a node (y, y) exists such that $d_{O_{R_n}}((x, x), (y, y)) = 2D(R_n)+1$. If f respects d_{R_n} and has at least one fixed-point, then the distance of at least one pair (x, x), (y, y) remains $2D(R_n) + 1$ in an Extended OTIS-R_n network.

Lemma 5.33 *If $f \in \mathcal{F}_{R_n}$, f respects d_{R_n} and $\{x \mid f(x) = x\} \neq \emptyset$, then $D(X_{R_n,f}) = 2D(R_n) + 1$.*

Proof:
There are $x, y \in V_n$ such that $f(x) = x$ and $d_{R_n}(x, y) = D(R_n)$. Let w

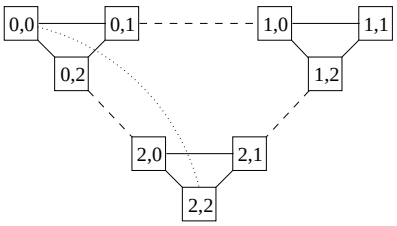

Figure 5.7: The structure of $X_{R_3,f}$.

This figure shows $X_{R_3,f}$, where $f(0) = 2$, $f(1) = 1$, and $f(2) = 0$. Note that $D(X_{R_3,f}) = 2D(R_3) + 1 = 3$. Optical links are depicted as dashed lines, f-links as dotted lines, and electronic links as solid lines.

be a shortest path between (x, x) and (y, y). If w uses no f-link, then $|w| = 2D(R_n)+1$ (Lemma 5.20). If w uses f-link $\{(z, z), (f(z), f(z))\}$, then

$$\begin{aligned}
|w| \geq\ & d_{O_{R_n}}((x,x),(z,z)) + d_{O_{R_n}}((y,y),(f(z),f(z))) + 1 \\
\geq\ & 2d_{R_n}(x,z) + 2d_{R_n}(y,f(z)) + 1 \\
=\ & 2d_{R_n}(f(x),f(z)) + 2d_{R_n}(y,f(z)) + 1 \\
\geq\ & 2d_{R_n}(f(x),y) + 1 \\
=\ & 2d_{R_n}(x,y) + 1 \\
=\ & 2D(R_n) + 1.
\end{aligned}$$

◇

Corollary 5.34 *For odd n there is no $f \in \mathcal{F}_{R_n}$ such that f respects d_{R_n} and $D(X_{R_n,f}) < 2D(R_n) + 1$.*

Proof:

For odd n a function $f \in \mathcal{F}_{R_n}$ has a fixed-point. ◇

For $x \in V_n$ and even n, we set

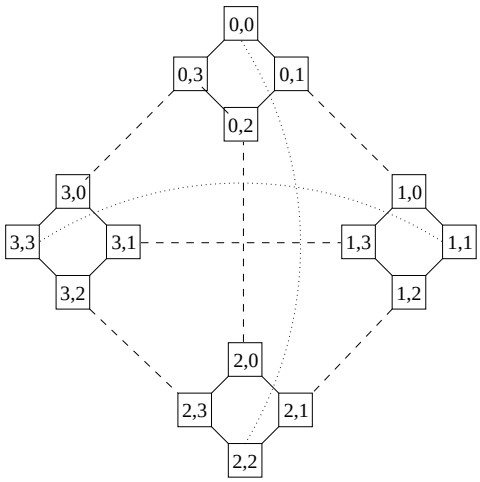

Figure 5.8: The structure of X_{R_4,f_4}.

Optical links are depicted as dashed lines, f-links as dotted lines, and electronic links as solid lines.

$$f_n(x) \stackrel{\text{def}}{=} (x + \tfrac{n}{2}) \bmod n = \begin{cases} x + \frac{n}{2} & , \quad x < \frac{n}{2} \\ x - \frac{n}{2} & , \quad x \geq \frac{n}{2}. \end{cases}$$

Figure 5.8 presents X_{R_4,f_4}. The following lemma shows that f_n respects d_{R_n}.

Lemma 5.35 *For even $n \geq 2$ the function $f_n : V_n \longrightarrow V_n$ has the following properties:*

1. $f_n \circ f_n = id_{V_n}$
2. *f_n respects d_{R_n}*
3. $\forall x, y \in V_n : d_{R_n}(x, f_n(y)) + d_{R_n}(x, y) = D(R_n)$.

Proof:

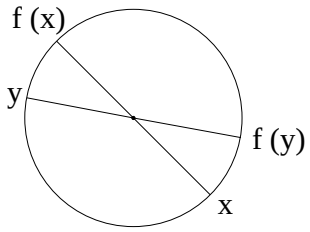

Figure 5.9: Mirror property of f_n.

1. Let $x \in V_n$. $f_n(f_n(x)) = (x + n) \bmod n = x$.

2. Let $x, y \in V_n$. If $x, y < \frac{n}{2}$ or $x, y \geq \frac{n}{2}$, then $|f_n(x) - f_n(y)| = |x - y|$ and hence $d_{R_n}(f_n(x), f_n(y)) = d_{R_n}(x, y)$. If $x < \frac{n}{2}$ and $y \geq \frac{n}{2}$ or $x \geq \frac{n}{2}$ and $y < \frac{n}{2}$, then $|f_n(x) - f_n(y)| = n - |x - y|$ and hence $d_{R_n}(f_n(x), f_n(y)) = d_{R_n}(x, y)$.

3. Let $x, y \in V_n$. Observe that f_n mirrors x at the centre of the ring (see Figure 5.9), i.e. $d_{R_n}(x, f_n(x)) = \frac{n}{2} = D(R_n)$. Hence $d_{R_n}(y, x) + d_{R_n}(y, f_n(x)) = D(R_n)$.

$\diamond$

For Extended OTIS-Rings Lemma 5.22 can be simplified. For these networks it is sufficient to calculate a minimum of four elements.

Lemma 5.36 *Let p, p', g, $g' \in V_n$, $S = \{p, f_n(p'), g, f_n(g')\}$, and δ_x defined as in Lemma 5.22. Then*

$$\min\{d_{R_n}(p, x) + d_{R_n}(g, x) + d_{R_n}(p', f_n(x)) + d_{R_n}(g', f_n(x)) + \delta_x \mid x \in V_n\}$$
$$=$$
$$n + \min\{d_{R_n}(p, x) + d_{R_n}(g, x) - d_{R_n}(p', x) - d_{R_n}(g', x) + \delta_x \mid x \in S\}$$

Proof:

For $x \in V_n$ we define

$$\begin{array}{rccl} g_x: & [n+2] & \longrightarrow & \mathbb{N}_0 \\ & y & \mapsto & d_{R_n}(x, y \bmod n) \end{array}$$

For all $y \in [n+1]$ it holds $g_x(y+1) - g_x(y) \in \{-1, 1\}$.

In the case $g_x(y+1) - g_x(y) = 1$ and $y \in [n+1]$, we say g_x *increase at* y and in the case $g_x(y+1) - g_x(y) = -1$ and $y \in [n+1]$, we say g_x *decrease at* y.

For $x > 0$, $y \in \{1, \ldots, n\}$, we get:

If g_x decrease at $y-1$ and increase at y, then $y = x$.

If g_0 decrease at $y-1$ and increase at y, then $y = n$.

For $\emptyset \neq X \subseteq V_n$ consider the function

$$\begin{array}{rccl} h_X: & [n+2] & \longrightarrow & \mathbb{N}_0 \\ & y & \mapsto & \sum\limits_{x \in X} g_x(y). \end{array}$$

For all $y \in [n+1]$ it holds $h_X(y+1) - h_X(y) \in \{-|X|, \ldots, |X|\}$. For $y \in [n+1]$ we say h_X *increase at* y if $h_X(y+1) - h_X(y) > 0$, h_X *is constant at* y if $h_X(y+1) - h_X(y) = 0$, and h_X *decrease at* y if $h_X(y+1) - h_X(y) < 0$.

Now we prove

$$\min\{h_X(y) \mid y \in \{1, \ldots, n\}\} = \min\{h_X(y) \mid y \in X\}. \tag{5.2}$$

If h_X is constant at y for all $y \in \{1, \ldots, n\}$, then 5.2 holds. If h is not constant for all $y \in \{1, \ldots, n\}$, then $y' \in \{1, \ldots, n\}$ exists such that $h_X(y') = \min\{h_X(y) \mid y \in \{1, \ldots, n\}\}$ and h_X increase at y' and h_X decrease at $y'-1$ or is constant at $y'-1$.

If function h_X increase at y' and h_X decrease at $y'-1$ or is constant at $y'-1$, then at least one function g_x, $x \in X$ decrease at $y'-1$ and increase at y'. So $y' \bmod n \in X$. Hence 5.2 holds.

Function f_n respects d_{R_n}. Therefore

$$d_{R_n}(p,x) + d_{R_n}(g,x) + d_{R_n}(p', f_n(x)) + d_{R_n}(g', f_n(x))$$
$$=$$
$$d_{R_n}(p,x) + d_{R_n}(g,x) + d_{R_n}(f_n(p'),x) + d_{R_n}(f_n(g'),x).$$

Setting $X = S$ in 5.2 provides

$$\min\{d_{R_n}(p,x) + d_{R_n}(g,x) + d_{R_n}(f_n(p'),x) + d_{R_n}(f_n(g'),x) \mid x \in V_n\}=$$
$$\min\{d_{R_n}(p,x) + d_{R_n}(g,x) + d_{R_n}(f_n(p'),x) + d_{R_n}(f_n(g'),x) \mid x \in S\}.$$

Lemma 5.35 yields $d_{R_n}(f_n(p'),x) = \frac{n}{2} - d_{R_n}(p',x)$ and $d_{R_n}(f_n(g'),x) = \frac{n}{2} - d_{R_n}(g',x)$. Furthermore, $\delta_x = 2$ for $x \in V_n - S$ and $\delta_x \leq 2$ for $x \in S$. $\diamond$

We show that the length of a shortest path is at most $\frac{n}{2}+2$. The proofs are based on exhaustive case distinction.

In the next two lemmata, we show that in some cases the length of a path that uses no f-link can be bound from above by $\frac{n}{2}+2$.

Lemma 5.37 *Let p, p', g, $g' \in V_n$, w_1 a shortest path between p and g in R_n, and w_2 a shortest path between p' and g' in R_n. If $V(w_1) \cap V(w_2) \neq \emptyset$, then $min\{d_{R_n}(g,g') + d_{R_n}(p,p') + 2, d_{R_n}(g,p') + d_{R_n}(p,g') + 1\} < \frac{n}{2} + 2$.*

Proof:

We set $a := d_{R_n}(g,g') + d_{R_n}(p,p') + 2$ and $b := d_{R_n}(g,p') + d_{R_n}(p,g') + 1$. If $a+b < n+4$, then $\min\{a,b\} < \frac{n}{2}+2$. The following three cases are possible.

Case w_1 uses p' and g'.

In this case $d_{R_n}(g,g') + d_{R_n}(p,p') = d_{R_n}(g,p') + d_{R_n}(g',p) = d(g,p)$. Hence $a + b \leq 2d_{R_n}(g,p) + 3 \leq n + 3$.

Case w_1 uses g' but not p'.

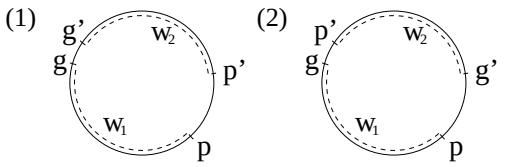

Figure 5.10: Cases for the proof of Lemma 5.38.

In this case $d_{R_n}(g,g') + d_{R_n}(g',p) = d_{R_n}(g,p)$. There are two subcases. **Subcase** w_2 **uses** p. There are two possibilities. First $n = d_{R_n}(p,p') + d_{R_n}(p,g') + d_{R_n}(g,g') + d_{R_n}(p',g)$ and second $d_{R_n}(p',g) = d_{R_n}(p,p') + d_{R_n}(p,g') + d_{R_n}(g,g')$. For both possibilities we get $a+b \leq n+3$. **Subcase** w_2 **uses** g. There are two possibilities. First $n = d_{R_n}(p,g') + d_{R_n}(g,g') + d_{R_n}(g,p') + d_{R_n}(p,p')$ and second $d_{R_n}(p,p') = d_{R_n}(p,g') + d_{R_n}(g',g) + d_{R_n}(g,p')$. For both possibilities we get $a+b = n+3$.

Case w_1 **uses** p' **but not** g'.

The proof can be done analogously to the case w_1 uses p' and g'. Write p' where g' occurs and vice versa. ◇

Lemma 5.38 *Let $p,p',g,g' \in V_n$, w_1 be a shortest path between p and g and w_2 a shortest path between p' and g' in R_n. If $V(w_1) \cap V(w_2) = \emptyset$ and $d_{R_n}(p,g) + d_{R_n}(p',g') \geq \frac{n}{2}$, then $min\{d_{R_n}(g,g') + d_{R_n}(p,p') + 2, d_{R_n}(g,p') + d_{R_n}(p,g') + 1\} < \frac{n}{2} + 3$.*

Proof:

We set $a := d_{R_n}(g,g') + d_{R_n}(p,p') + 2$ and $b := d_{R_n}(g,p') + d_{R_n}(p,g') + 1$. There are two possible cases. See Figure 5.10. In case 1, we have $d_{R_n}(p,g) + d_{R_n}(g,g') + d_{R_n}(g',p') + d_{R_n}(p',p) = n$. So $d_{R_n}(p,g) + d_{R_n}(p',g') + a = n+2$ and hence $a \leq \frac{n}{2} + 2$ follows.

In case 2, we have $d_{R_n}(p,g) + d_{R_n}(g,p') + d_{R_n}(p',g') + d_{R_n}(g',p) = n$. So

$d_{R_n}(p,g) + d_{R_n}(p',g') + b = n+1$ and hence $b \leq \frac{n}{2}+1$ follows. $\diamond$

Now we show that if a path with no f-link has a length greater than or equal to $\frac{n}{2}+3$, then there exists a path that uses one f-link and has a length of at most $\frac{n}{2}+2$.

Lemma 5.39 *Let p, p', g, $g' \in V_n$, $S := \{p, f_n(p'), g, f_n(g')\}$, w_1 a shortest path between p and g in R_n, and w_2 a shortest path between p' and g' in R_n. If $V(w_1) \cap V(w_2) = \emptyset$, $d_{R_n}(p,g) + d_{R_n}(p',g') < \frac{n}{2}$ and $min\{d_{R_n}(g,g') + d_{R_n}(p,p') + 2, d_{R_n}(g,p') + d_{R_n}(p,g') + 1\} \geq \frac{n}{2}+3$, then $n + \min\{d_{R_n}(p,x) + d_{R_n}(g,x) - d_{R_n}(p',x) - d_{R_n}(g',x) + \delta_x \mid x \in S\} + 1 < \frac{n}{2}+3$.*

Proof:

We set $a := d_{R_n}(p,g)$, $b := d_{R_n}(p,p')$, $c := d_{R_n}(p,g')$, $d := d_{R_n}(g,p')$, $e := d_{R_n}(g,g')$, and $f := d_{R_n}(p',g')$. Let $x_1 := n - b - d + f + \delta_{f(p')} + 1$, $x_2 := n - c - e + f + \delta_{f(g')} + 1$, $x_3 := n - b - c + a + \delta_p + 1$, $x_4 := n - d - e + a + \delta_g + 1$, and $x_5 := \min\{x_1, x_2, x_3, x_4\}$. Observe that $x_5 = n + \min\{d_{R_n}(p,x) + d_{R_n}(g,x) - d_{R_n}(p',x) - d_{R_n}(g',x) + \delta_x \mid x \in S\} + 1$, $a + f < \frac{n}{2}$, $e + b \geq \frac{n}{2}+1$, and $d + c \geq \frac{n}{2}+2$. There are twelve possible positions for p, p', g, g'. See Figure 5.11.

This results in four cases:

1. $e = d + f$: $x_2 = n - (c+d) + \delta_{f(g')} + 1 \leq \frac{n}{2} + \delta_{f(g')} - 1 \leq \frac{n}{2}+1$.
2. $d = e + f$: $x_1 = n - (b+e) + \delta_{f(p')} + 1 \leq \frac{n}{2} + \delta_{f(p')} \leq \frac{n}{2}+2$.
3. $d = a + b$: $x_4 = n - (b+e) + \delta_g + 1 \leq \frac{n}{2} + \delta_g \leq \frac{n}{2}+2$.
4. $e = a + c$: $x_4 = n - (c+d) + \delta_p + 1 \leq \frac{n}{2} + \delta_p - 1 \leq \frac{n}{2}+1$.

$\diamond$

Now we consider the case $n = 2$.

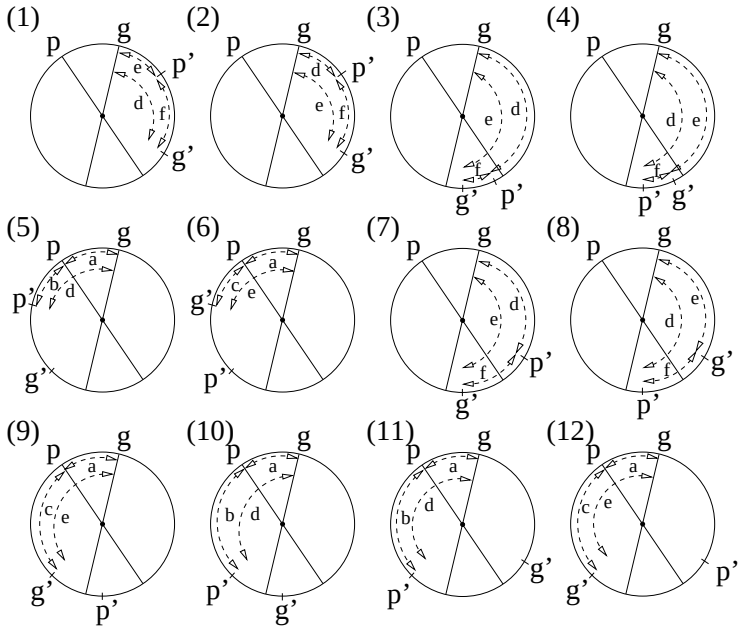

Figure 5.11: Cases for the proof of Lemma 5.39.

Lemma 5.40 *$D(X_{R_2,f_2}) = 2$ and for all $f \in \mathcal{F}_{R_2}$: $D(X_{R_2,f}) \geq 2$.*

Proof:

See Figure 5.5. ◇

Before we calculate the diameter for the Extended OTIS-R_n network using f_n, we give a lower bound for the diameter.

Theorem 5.41 *Let $n \in \mathbb{N}$. For all $f \in \mathcal{F}_{R_n}$: $D(X_{R_n,f}) \geq \frac{n}{2} + 2$.*

Proof:

The case $n = 2$ follows from Lemma 5.40.

For $n > 2$ the proof is similar to the proof of Theorem 5.29, case $n \bmod 3 = 2$. We omit the details and only give s, t, s_1 and t_1.

Case $n = 4m$ for $m > 0$: $s = (0, m)$, $t = (3m, 2m)$, $s_1 = (0, 3k)$, $t_1 = (k, 2k)$.

Case $n = 4m+2$ for $m > 0$: $s = (0, 2m+2)$, $t = (2m, 2m+1)$, $s_1 = (0, 2m)$, $t_1 = (2m + 2, 2m + 1)$. ◇

Corollary 5.42 *For even $n > 0$ any addition of links to the OTIS-R_n network that does not increase the degree of the network results in a diameter $\geq \frac{n}{2} + 2$.*

Now we are able to conclude that for all even $n > 0$ the diameter of the Extended OTIS-R_n network using f_n is $D(R_n) + 2$.

Theorem 5.43 (Diameter of $X_{R_n,f}$.) *For even $n > 0$ the diameter of the Extended OTIS-R_n network using f_n is $\frac{n}{2} + 2 = D(R_n) + 2$.*

Proof:

By Theorem 5.41 $D(X_{R_n,f}) \geq D(R_n) + 2$.

Let (p, g), (p', g') be two nodes of X_{R_n,f_n}. By Lemma 5.39, 5.38, 5.37, and 5.36:

$$\min\{(a),(b),(c)\} \leq \tfrac{n}{2}+2, \text{ where}$$

(a) $= 1 + d_{R_n}(g',p) + d_{R_n}(g,p')$,

(b) $= 2 + d_{R_n}(p,p') + d_{R_n}(g,g')$, and

(c) $= \min\{d_{R_n}(g,x) + d_{R_n}(p,x) + d_{R_n}(g',f(x)) + d_{R_n}(p',f(x)) + \delta_x \mid x \in V_n, f(x) \neq x\} + 1$

Theorem 5.26 yields $D(X_{R_n,f}) \leq \frac{n}{2}+2$. Note that $\frac{n}{2}+2 = D(R_n)+2$.

$\diamond$

5.7.3.3 One-dimensional Meshes.

For $x \in [n]$, we set

$$f_n(x) \stackrel{\text{def}}{=} n - x - 1.$$

Function f_n respects $d_{M_{1,n}}$. Figures 5.12, 5.13 present $X_{M_{1,4},f_4}$ and $X_{M_{1,5},f_5}$. For one-dimensional meshes a lemma similar to Lemma 5.36 for rings can be obtained and used to show that the diameter of an Extended OTIS-$M_{1,n}$ network using f_n is n. We present another way to proof this. The proof uses the fact that in $X_{M_{1,n},f_n}$ an subgraph exists that is isomorphic[5] to $X_{M_{1,n-2},f_{n-2}}$. For example, the subgraph $(V_{X_{M_{1,5},f_5}} \cap \{1,2,3\}^2, E_{X_{M_{1,5},f_5}} \cap \mathcal{P}_2(\{1,2,3\}^2))$ of $X_{M_{1,5},f_5}$ is isomorphic to $X_{M_{1,3},f_3}$ and the subgraph $(V_{X_{M_{1,4},f_4}} \cap \{1,2\}^2$, $E_{X_{M_{1,4},f_4}} \cap \mathcal{P}_2(\{1,2\}^2)$ of $X_{M_{1,4},f_4}$ is isomorphic to $X_{M_{1,2},f_2}$.

Lemma 5.44 *Let $n \in \mathbb{N}$, $n > 3$. The subgraph $(V_{X_{M_{1,n},f_n}} \cap \{1,\ldots,n-2\}^2, E_{X_{M_{1,n},f_n}} \cap \mathcal{P}_2(\{1,\ldots,n-2\}^2)$ of $X_{M_{1,n},f_n}$ is isomorphic to $X_{M_{1,n-2},f_{n-2}}$.*

[5] If graph G' is isomorphic to graph G via an isomorphism f, then $X_{G',f'}$ is isomorphic to $X_{G,f\circ f'\circ f^{-1}}$.

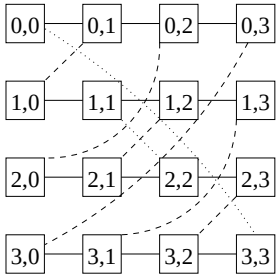

Figure 5.12: The structure of $X_{M_{1,4},f_4}$.

Optical links are shown as dashed lines, f-links as dotted lines, and electronic links as solid lines.

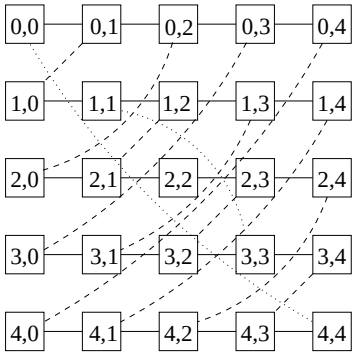

Figure 5.13: The structure of $X_{M_{1,5},f_5}$.

Optical links are shown as dashed lines, f-links as dotted lines, and electronic links as solid lines.

Proof:
Let $X = (V_{X_{M_{1,n},f_n}} \cap \{1,\ldots,n-2\}^2, E_{X_{M_{1,n},f_n}} \cap \mathcal{P}_2(\{1,\ldots,n-2\})$. The isomorphism maps node (x,y) of X to $(x-1,y-1)$ of $X_{M_{1,n-2},f_{n-2}}$. $\diamond$

Theorem 5.45 *The diameter of the Extended OTIS-$M_{1,n}$ network using f_n is n.*

Proof:
$D(X_{M_{1,n},f_n}) \geq n$ follows from Theorem 5.27. We prove the theorem by induction on $n \geq 2$ using Lemma 5.44. For $n = 2$, see Figure 5.5. For $n = 3$, see Figure 5.12. Now we assume that $n > 3$. We denote the subgraph $(V_{X_{M_{1,n},f_n}} \cap \{1,\ldots,n-2\}^2, E_{X_{M_{1,n},f_n}} \cap \mathcal{P}_2(\{1,\ldots,n-2\}^2))$ of $X_{M_{1,n},f_n}$ as *core* and the rest of the graph as *border*. By induction hypothesis a shortest path within the core has length $\leq n-2$. So we only have to observe paths from the border to the border and from the border to the core. Let $M := M_{1,n}$ and $X := X_{M,f_n}$. We use the results of Theorem 5.17 and Lemma 5.22 for the construction. First, we consider paths from border to border. Let $p, g \in [n]$.

1. A path between $(0,p)$ and $(g,0)$: Use path $p_1((0,p),(g,0))$ (Theorem 5.17). Path $p_1((0,p),(g,0))$ has a length $\leq n$. So $d_M((0,p),(g,0)) \leq n$.

2. A path between $(0,p)$ and $(g,n-1)$: For $g < p$ use path $p_2((0,p),(g,n-1))$ (Theorem 5.17). For $g \geq p$ use $p(\ (0,p)\ ,(g,n-1)\ ,(0,0))$ (Lemma 5.22). In both cases the path has at most length n. Hence $d_M\ ((0,p)\ ,(g,n-1)) \leq n$.

3. A path between $(0,p)$ and $(n-1,g)$: For $g < p$ use path $p_1((0,p),(n-1,g))$. For $g \geq p$ use $p((0,p),(n-1,g),(0,0))$ (Lemma 5.22). In both cases the path has at most length n. Hence $d_M((0,p),(n-1,g)) \leq n$.

4. A path between $(0,p)$ and $(0,g)$: A shortest path in group 0. Thus $d_M((0,p),(0,g)) \leq n$.

5. A path between $(p,0)$ and $(n-1,g)$: For $g < p$ use path $p_2((p,0),(n-1,g))$. For $g \geq p$ use $p((p,0),(n-1,g),(0,0))$ (Lemma 5.22). In both cases the path has at most length n. Hence $d_M((p,0),(n-1,g)) \leq n$.

6. A path between $(p,0)$ and $(g,n-1)$: For $g \leq p$ use $p_1((p,0),(g,n-1))$, for $g = p+1$ use $p((p,0),(g,n-1),(p,p))$ and for $g > p+1$ use $p((p,0),(g,n-1),(0,0))$. Thus $d_M((p,0),(g,n-1)) \leq n$.

7. A path between $(p,0)$ and $(g,0)$: The case $p = 0$ and $g = n-1$ is covered by the above cases. For $p = n-1$ and $g = 0$ use $p1((n-1,0),(0,0))$. For all other cases use $p2((p,0),(g,0))$. Hence $d_M((p,0),(g,0)) \leq n$.

8. A path between $(n-1,p)$ and $(n-1,g)$: A shortest path in group $n-1$. Thus $d_M((n-1,p),(n-1,g)) \leq n$.

9. A path between $(n-1,p)$ and $(g,n-1)$: Use $p_1((n-1,p),(g,n-1))$. Hence $d_M((n-1,p),(g,n-1)) \leq n$.

10. A path between $(p,n-1)$ and $(g,n-1)$: Cases $p = n-1$ and $g = 0$ and $p = 0$ and $g = n-1$ are covered by the above cases. For all other cases use $p2((p,n-1),(g,n-1))$. Thus $d_M((p,n-1),(g,n-1)) \leq n$.

Now we consider the case border to core. All nodes of the border without the nodes $\{(0,0),(0,n-1),(n-1,0),(n-1,n-1)\}$ have distance of at most two from the core and hence for these nodes the result follows by induction. So we have to consider shortest paths between nodes of $\{(0,0),(0,n-1),(n-1,0),(n-1,n-1)\}$ and the core. Let $i,j \in \{1,\ldots,n-2\}$. If we do not mention that a f-link is used, then a path using no f-link has length $\leq n$.

1. $d_X((0,0),(i,j)) \leq n$. For $i+j > n-1$ use the f-link at $(0,0)$.
2. $d_X((0,n-1),(i,j)) \leq n$.
3. $d_X((n-1,0),(i,j)) \leq n$.
4. $d_X((n-1,n-1),(i,j)) \leq n$. For $i+j < n-1$ use the f-link at $(n-1,n-1)$.

$\diamond$

Corollary 5.46 *The function f_n is optimal for the diameter of the Extended OTIS-$M_{1,n}$ network using f, i.e. for all $f \in \mathcal{F}_{M_{1,n}}$: $D(X_{M_{1,n},f}) \geq D(X_{M_{1,n},f_n})$.*

Proof:
$D(X_{M_{1,n},f}) \geq n$ follows from Theorem 5.27. $\diamond$

5.7.3.4 Two-dimensional Meshes.

For $(x,y) \in [n] \times [n]$, we set

$$f_n((x,y)) \stackrel{\text{def}}{=} (n-x-1, n-y-1).$$

Function f_n respects $d_{M_{2,n}}$. Figure 5.6 shows an Extended OTIS-$M_{2,2}$ network using f_2 and figure 5.14 shows an Extended OTIS-$M_{2,3}$ using f_3.

As in the case of one-dimensional meshes an subgraph of $X_{M_{2,n},f_n}$ exists that is isomorphic to $X_{M_{2,n-2},f_{n-2}}$.

Lemma 5.47 *For $n > 3$ the subgraph $(V_{X_{M_{2,n},f_n}} \cap \{1,\ldots,n-2\}^4, E_{X_{M_{1,n},f_n}} \cap \mathcal{P}_2(\{1,\ldots,n-2\}^4))$ of $X_{M_{2,n},f_n}$ is isomorphic to $X_{M_{2,n-2},f_{n-2}}$.*

With the help of the above lemma, we prove $D(X_{M_{2,n},f_n}) = 2n$.

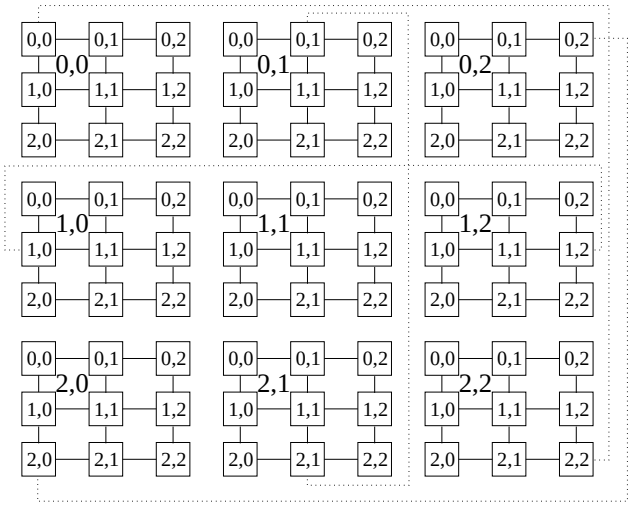

Figure 5.14: The structure of $X_{M_{2,3},f_3}$ (without optical links).

Only electronic and f-links are shown.

Theorem 5.48 *The diameter of the Extended OTIS-$M_{2,n}$ network using f_n is $2n$.*

Proof:

The proof that $D(X_{M_{2,n},f_n}) \leq 2n$ can be done analogously to the one-dimensional case and is omitted.

For $s = ((0,0),(1,0))$ and $t = ((0,1),(n-1,n-1))$ we have $d_{X_{M_{2,n},f_n}}(s,t) = 2n$ and hence $D(X_{M_{2,n},f_n}) = 2n$. ◇

Theorem 5.49 *The diameter of the Extended OTIS-$M_{2,n}$ network using an $f \in \mathcal{F}_{M_{2,n}}$ is greater than or equal to $2n$.*

Proof:

The proof is similar to the proof of Theorem 5.29, case $n \bmod 3 = 2$. We omit the details.

We give s, t, s_1 and t_1: $s = ((0,0),(1,0))$, $t = ((0,1),(n-1,n-1))$, $s_1 = ((0,0),(0,1))$, $t_1 = ((1,0),(n-1,n-1))$. $\diamond$

Corollary 5.50 *The function f_n is optimal for the diameter of the Extended OTIS-$M_{2,n}$ network using f, i.e. for all $f \in \mathcal{F}_{M_{2,n}}$: $D(X_{M_{2,n},f}) \geq D(X_{M_{2,n},f_n})$.*

5.8 Conclusion.

In this chapter we investigated aspects of routing in OTIS-G networks. We gave lower bounds for k-k routing and sorting algorithms in these networks and succeed in constructing a fast k-k sorting algorithm with a small buffer size for the OTIS-Mesh. We did not succeed in matching the bisection or diameter lower bound for the OTIS-Mesh. Further research is necessary to improve the lower or upper bound. We further showed that OTIS-G networks are well suited for oblivious k-k routing, i.e., an oblivious k-k routing algorithm with an asymptotically optimal running time and a buffer size of $O(1)$ exists for these networks, provided G is of fixed degree. We determined the diameter of OTIS-G networks. This led to the definition of Extended OTIS-G networks, where we reduced the diameter of an OTIS-G network by adding links such that the degree of the network is not increased. We succeed in giving optimal extensions for hypercubes, rings of even size, one-dimensional meshes, and two-dimensional meshes.

Bibliography

[1] S. N. Bhatt, F. R. K. Chung, J. W. Hong, F. T. Leighton, and A. L. Rosenberg. Optimal Simulations by Butterfly Networks. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 192–204, 1988.

[2] A. Borodin and J. E. Hopcroft. Routing, Merging, and Sorting on Parallel Models of Computation. *Journal of Computer and System Sciences*, 30(1):130–145, February 1985.

[3] A. Borodin, P. Raghavan, B. Schieber, and E. Upfal. How much can Hardware help Routing? In *Proceedings of the 25th Annual ACM Symposium on the Theory of Computing*, pages 573–582, 1993.

[4] A. Z. Broder, A. M. Frieze, and E. Upfal. A General Approach to Dynamic Packet Routing with Bounded Buffers. *Journal of the ACM*, 48(2):324–349, 2001.

[5] K. Day and A. Al-Ayyoub. Topological Properties of OTIS-Networks. *IEEE Transactions on Parallel and Distributed Systems*, (accepted 2001).

[6] M. Feldman, S. Esener, C. Guest, and S. H. Lee. Comparison Between Optical and Electrical Interconnection Based on Power and Speed Considerations. *Applied Optics*, 27(9):1742–1751, May 1988.

[7] M. D. Grammatikakis, D. F. Hsu, and J. F. Sibeyn. Packet Routing in Fixed-Connection Networks: A Survey. *Journal of Parallel and Distributed Computing*, 54(2):77–132, November 1998.

[8] K. Iwama, Y. Kambayashi, and E. Miyano. New Bounds for Oblivious Mesh Routing. *Lecture Notes in Computer Science*, 1461:295–306, 1998.

[9] K. Iwama, Y. Kambayashi, and E. Miyano. New Bounds for Oblivious Mesh Routing. *Journal of Graph Algorithms and Applications*, 5(5):17–38, 2001.

[10] K. Iwama and E. Miyano. Three-Dimensional Meshes are Less Powerful than Two-Dimensional Ones in Oblivious Routing. *Lecture Notes in Computer Science*, 1284:284–295, 1997.

[11] K. Iwama and E. Miyano. An O($\sqrt{N}$) Oblivious Routing Algorithm for 2-D Meshes of Constant Queue-Size. In *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 466-475,1999.

[12] K. Iwama and E. Miyano. A (2.954 + e)n Oblivious Routing Algorithm on 2D Meshes. In *Proceedings of the 12th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 186–195, 2000.

[13] K. Iwama and E. Miyano. A Lower Bound for Elementary Oblivious Routing on Three-Dimensional Meshes. *Journal of Algorithms*, 39(2):145–161, May 2001.

[14] K. Iwama and E. Miyano. An O($\sqrt{N}$) Oblivious Routing Algorithm for 2-D Meshes of Constant Queue-Size. *Journal of Algorithms*, 41(2):262–279, November 2001.

[15] N. Kahale and F. T. Leighton. Greedy Dynamic Routing on Arrays. *Journal of Algorithms*, 29(2):390–410, November 1998.

[16] C. Kaklamanis, D. Krizanc, and T. Tsantilas. Tight Bounds for Oblivious Routing in the Hypercube. In *Proceedings of the 2nd Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 31–36, 1990.

[17] C. Kaklamanis, D. Krizanc, and T. Tsantilas. Tight Bounds for Oblivious Routing in the Hypercube. *Mathematical Systems Theory*, 24(4):223–232, 1991.

[18] J. Karaganis. On the Cube of a Graph. *Canadian Mathematics Bulletin*, 11:295–296, 1969.

[19] M. Kaufmann, J. F. Sibeyn, and T. Suel. Derandomizing Algorithms for Routing and Sorting on Meshes. In *Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 669–679, 1994.

[20] D. Knuth. *The Art of Computer Programming, Vol. III: Sorting and Searching*. Addison-Wesley, Reading, MA, 1973.

[21] R. Kock, F. T. Leighton, B. Maggs, S. Rao, and A. Rosenberg. Work-Preserving Emulations of Fixed-Connection Networks. In *Proceedings of the 21th Annual ACM Symposium on Theory of Computing*, pages 227–240, 1989.

[22] D. Krizanc. Oblivious Routing with Limited Buffer Capacity. *Journal of Computer and System Sciences*, 43(2):317–327, October 1991.

[23] D. Krizanc, D. Peleg, and E. Upfal. A Time-Randomness Tradeoff for Oblivious Routing. In *Proceedings of the 20th Annual ACM Symposium on the Theory of Computing*, pages 93–102, 1988.

[24] M. Kunde. Optimal Sorting on Multi-Dimensionally Mesh-Connected Computers. In *Proceedings of the 4th Annual Symposium on Theoretical Aspects of Computer Science*, pages 408–419, 1987.

[25] M. Kunde. Block Gossiping on Grids and Tori: Deterministic Sorting and Routing match the Bisection Bound. In *Proceedings of the First Annual European Symposium (ESA'93)*, pages 272–283. LNCS 726. 1993.

[26] M. Kunde, R. Niedermeier, K. Reinhardt, and P. Rossmanith. Optimal Average Case Sorting on Arrays. In *Proceedings of the 12th Annual Symposium on Theoretical Aspects of Computer Science*, pages 503–514. LNCS 900. 1995.

[27] M. Kunde, R. Niedermeier, and P. Rossmanith. Faster Sorting and Routing on Grids with Diagonals. In *Proceedings of the 11th Annual Symposium on Theoretical Aspects of Computer Science*, pages 225–236. LNCS 775. 1994.

[28] M. Kunde, R. Niedermeyer, K. Reinhardt, and P. Rossmanith. Optimal Deterministic Sorting and Routing on Grids and Toris with Diagonals. *Algorithmica*, 25:438–458, 1999.

[29] M. Kunde and T. Tensi. Bounds for Oblivious Message Routing on Grids. In *Research in Informatics: Proceedings of Parcella 90*, 2:227–238, 1990.

[30] F. T. Leighton. Tight Bounds on the Complexity of Parallel Sorting. *IEEE Transaction on Computers*, 34(4):344–354, 1985.

[31] F. T. Leighton. *Introduction to Parallel Algorithms and Architectures.* Morgan Kaufman, San Mateo, 1992.

[32] A. Litman and S. Moran-Schein. Fast, Minimal and Oblivious Routing Algorithms on the Mesh with Bounded Queues. In *Proceedings of the 13th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 21–31, 2001.

[33] P. Marchand, A. Krishnamoorthy, F. Kiamilev, and S. Esener. Grain-size Considerations for Optoelectronic Multistage Interconnection Networks. *Applied Optics*, 31(26):5480–5507, 1992.

[34] P. Marchand, F. Zane, R. Paturi, and S. Esener. Scalable Network Architectures using the Optical Transpose Interconnection System (OTIS). *Journal of Parallel and Distributed Computing*, 60(5):521–538, 2000.

[35] G. C. Marsden, P. J. Marchand, P. Harvey, and S. C. Esener. Optical Transpose Interconnection System Architectures. *Optics Letters*, 18(13):1083–1085, 1993.

[36] O. Melnikow, R. Tyshkevich, V. Yemelichev, and V. Sarvanov. *Lectures on Graph Theory.* Wissenschaftsverlag, 1994.

[37] F. Meyer auf der Heide and C. Scheideler. Communication in Parallel Systems. *Lecture Notes in Computer Science*, 1175:16–34, 1996.

[38] B. Monien and H. Sudborough. Embedding one Interconnection Network in Another. *Computing Supplementum*, 7:257–282, 1990.

[39] A. Osterloh. Sorting on the OTIS-Mesh. In *Proceedings on the 14th International Parallel and Distributed Processing Symposium*, pages 269–274, 2000.

[40] A. Osterloh. Oblivious Routing on d-Dimensional Meshes. In *Proceedings of the 8th International Colloquium on Structural Information and Communication Complexity*, pages 305–320, 2001.

[41] I. Parberry. An Optimal Time Bound for Oblivious Routing. *Algorithmica*, 5:243–250, 1990.

[42] S. Rajasekaran and S. Sahni. Randomized Routing, Selection, and Sorting on the OTIS-Mesh. *IEEE Transactions on Parallel and Distributed Systems*, 9(9):833–840, September 1998.

[43] S. Rajasekaran and T. Tsantilas. Optimal Routing Algorithms for Mesh-Connected Processor Arrays. *Algorithmica*, 8:21–38, 1992.

[44] C. Scheideler. Universal Routing Strategies for Interconnection Networks. *Lecture Notes in Computer Science 1390*, 1998.

[45] M. Sekanina. On an Ordering of the Set of Vertices of a Connected Graph. *Publications of the Faculty of Science, University of Brno*, 412:137–142, 1960.

[46] J. F. Sibeyn, B. S. Chlebus, and M. Kaufmann. Deterministic Permutation Routing on Meshes. *Journal of Algorithms*, 22(1):111–141, January 1997.

[47] J.F. Sibeyn. Algorithms for Routing on Meshes. Ph. d. thesis, Computer Science Department, Utrecht University, 1992.

[48] T. Suel. Improved Bounds for Routing and Sorting on Multi-Dimensional Meshes. In *Proceedings of the 6th ACM Symposium on Parallel Algorithm and Architectures*, pages 26–35, 1994.

[49] L. G. Valiant. A scheme for fast parallel communication. *SIAM Journal on Computing*, 11(2):350–361, May 1982.

[50] C. Wang and S. Sahni. BPC Permutations on the OTIS-Mesh Optoelectronic Computer. In *Proceedings on the 4th International Conference on Massively Parallel Processing Using Optical Interconnections*, pages 130–135, 1997.

[51] C. Wang and S. Sahni. Basic Operations on the OTIS-Mesh Optoelectronic Computer. *IEEE Transactions on Parallel and Distributed Systems*, 9(12):1226–1236, 1998.

[52] C. Wang and S. Sahni. BPC Permutations on the OTIS-Hypercube Optoelectronic Computer. *Informatica*, 22(3):263–269, 1998.

[53] C. Wang and S. Sahni. OTIS Optoelectronic Computer. In *Parallel Computation Using Optical Interconnections*, chapter 5. Kluwer Academic Publishers, 1998.

[54] C. Wang and S. Sahni. Matrix Multiplication on the OTIS-Mesh Optoelectronic Computer. In *Proceedings on the 5th International Conference on Parallel Interconnects*, pages 131–138, 1999.

[55] C. Wang and S. Sahni. Image Processing on the OTIS-Mesh Optoelectronic Computer. *IEEE Transactions on Parallel and Distributed Systems*, 11(2):97–109, February 2000.

[56] D.B. West *Introduction to Graph Theory.* Prentice Hall, Second Edition, 2001.

[57] S.K. Yun and K. H. Park. Comments on "Hierarchical Cubic Networks". *IEEE Transactions on Parallel and Distributed Systems*, 9(4):410-414, April 1998.

[58] F. Zane, P. J. Marchand, R. Paturi, and S. C. Esener. Sacalable Network Architectures Using the Optical Transpose Interconnection Sys-

tems (OTIS). In *Proceedings on the Fourth International Conference on Massively Parallel Processing Using Optical Interconnections*, pages 114–121, 1996.

Index

$((S_i),(D_i),(entry_i),(T_i))$, 56
(V,E), 7
(V,E,r), 9
$(V_i,E_i,exit_i)$, 56
(α,β,γ), 28
(j,c,i), 22
(x,y), 9
$(\mathcal{P},src,dst)$, 10
$< a,b,c,d,i,j >$, 94
$B(a,b,c,d)$, 94
$C_G(X,X')$, 8
$D(G)$, 8
G_g, 83
H_n, 123
I_i, 28
$M(i,j)$, 94
$M_{r,n}$, 16
O_G, 82
R_n, 126
$SB(a,b)$, 94
$T_{l,i}$, 31
U_j, 32, 35
$V(p)$, 8
$X_{G,f}$, 115
$[a,b]$, 18
$[n]$, 7
Φ, 14
$\bar{\imath}$, 31
$\langle\rangle$, 30
δ, 31
η_i, 33, 36
$\longleftrightarrow_e$, 83, 115
$\longleftrightarrow_f$, 115
$\longleftrightarrow_o$, 83, 115
$\bar{\jmath}$, 32, 35
θ_j, 32, 35
ε, 87
$\vec{E}$, 9
$\vec{G}$, 9
$\mathbb{Z}$, 7
$\{x,y\}$, 8
$bw(G)$, 8
$d : \mathbb{N}_0 \longrightarrow \mathbb{N}$, 19
$d_G(x,y)$, 8, 9
$db(p)$, 59
$deg(G)$, 8

$deg_G(x)$, 8
$dst(p)$, 11
$entry_i$, 56
$exit_i$, 56
$f^{-1}(U)$, 7
f_n, 124, 127, 136, 140
$n_{i,j}$, 59
$p(s,t)$, 113
$p(s,t,X)$, 117
$p \leq_c q$, 22
$p_1(s,t)$, 113
$p_2(s,t)$, 113
$p_{i,j}$, 59
r_j, 32, 35
$rank(p, \leq_c)$, 22
s_1, 57
s_2, 57
s_3, 57
s_j, 32, 35
$sb(p)$, 59
$src(p)$, 11
$x \longleftrightarrow_g^* y$, 107, 116
$x \longleftrightarrow^* y$, 116
$x \longleftrightarrow_{-f}^* y$, 116
$x \Longrightarrow^* y$, 67
$x \Longrightarrow^* y$, 57
x_j, 35
y_j, 35
$z_{c,i}$, 21
$\mathcal{C}_x$, 19
$\mathcal{F}_G$, 122
$\mathcal{N}$, 9
$\mathcal{N}_{D,i}$, 56
$\mathcal{N}_{S,i}$, 56
$\mathcal{P}_2(M)$, 7
$\mathcal{P}_x$, 19
$\mathbb{N}$, 7
$\mathbb{N}_0$, 7
$^g ata_B(j)$, 101
$|\cdot|$, 7

address, 11
algorithm
  deterministic, 11
  for a problem, 13
  number of steps, 11
  routing
    k-k, 13
    buffer size, 12
    oblivious, 3, 13, 46
    pure, 47
  running time, 12
    bisection bound, 12
    diameter bound, 12
    lower bound, 12, 20
  simplicity, 12

block

destination, 57
source, 57
blocks, 87
border, 138
brick
dirty, 88
in OTIS-Mesh, 93
bricks, 87
buffer, 10
injection, 12
internal, 12
link, 12

communication link, 9
bandwidth, 10
electronic, 82, 115
inter-group, 83
intra-group, 82
optical, 83, 115
connection path, 57
core, 138
counter, 21
cycle, *see* graph

destination, 11
address, 10
node, 11
processor, 11
directed version, 9
dynamic routing, 30

embedding
$M_{4,n}$
into OTIS-$M_{2,n}$, 91
congestion, 15
dilation, 15
dynamic, 14
load, 15
one-dimensional mesh
into a network, 19
static, 14
empty step, 26

family of fixed degree, 17
flit, 10

gap routing, 20, 45
graph
bisection width, 8
complete, 2
connected, 8, 9
cycle, 9
degree, 8
diameter, 8, 9
directed, 8
directed edge, 8
incident, 9
directed version, 9
edge, 7
incident, 8
undirected, 7

family of fixed degree, 8
node, 7
adjacent, 8
degree, 8
distance, 8, 9
neighbour, 8
path, *see* path
length, 8
regular, 8
size, 8
tree, 9
undirected, 7

hierarchical cubic netwwork, 124
hypercube, 123

incident, 8
indexing, 86
continuous, 89
in OTIS-Mesh, 95
injector, 28
(α, β, γ), 28
$\bar{\imath}$, 31
δ, 31
η, 33, 36–38
creation rate, 28
disabled, 33
enabled, 29, 33
normal, 35
of U_j, 32, 35
restricted with offset, 29
set of bounded, 29
special, 36
internal computation, 10
interval, 17, 31
length, 31
step, 31
inverse image, 7

layer, 86
link, *see* communication link
local memory, 10

mesh
r-dimensional, 16
edge of i-th dimension, 16
edge set, 16
node set, 16
side length, 16
one-dimensional
size, 16
mesh-connected network, *see* mesh
model of computation, 9

network, 9
(t_1, t_2)-partitioning, 56
$M_{r,n}$, $r = 2$, 68
$M_{r,n}$, $r = 3$, 72
$M_{r,n}$, $r > 2$, r even, 71
$M_{r,n}$, $r > 3$, r odd, 74

OTIS-G, 105
size, 57

OTIS
G, 82
electronic links, 82
group, 83
inter-group links, 83
intra-group links, 82
optical links, 83
Extended, 115
f-link, 115
electronical link, 115
Hypercube, 123
One-dimensional mesh, 136
optical link, 115
optimal diameter, 123
Ring, 126
Two-dimensional Mesh, 140
Mesh, 83
block, 93
brick, 93
indexing of blocks, 95
indexing of meshes, 95
indexing of superblocks, 95
superblock, 93

packet, 10
class, 19
continuous sequence, 32
field, 11
additional information, 11
address, 11
message, 11
size, 11
layer, 86
pahse 2, 59
phase 1, 59
rank, 22
transportable, 21
path, 8, 9
connection, 57
cycle, 8
elementary, 49
Hamiltonian, 8
length, 8
set of nodes used by, 8
simple, 8
strongly-dimensional, 49
use edge, 8
uses f-links, 116
uses optical links, 107
weakly-dimensional, 49
place, 86
PRAM, 1
processing unit, 10
processor, 9

f respects d_G, 118

ring, 126
root, 9
route
    from v to v', 58
    on a cycle, 58
routing model
    cut-through, 10
    relaxed, 49
    store-and-forward, 10
    wormhole, 10
routing problem
    k-k, 2, 11
    dynamic, 18, 28, 45
    full, 11
    instance, 13
    packet routing problem, 10
    permutation, 2, 11
    static, 18

simulation
    one-dimensional mesh
      by a network, 19
    slowdown, 15
sorting
    0-1 principle, 88
    with all-to-all mappings, 87
sorting problem, 86
    k-k, 86
    permutation, 86
source, 11
    node, 11
    processor, 11
step, 10
synchronous, 9

time difference, 19

virtual node, 20

Theses of the dissertation.

1. Efficient data transport in parallel computers build on sparse interconnection networks is crucial for their performance.

2. A basic data transport problem in such a computer is the k-k routing problem, in which each processor (computer) sends and receives at most k packets.

3. The family of mesh-connected networks is one of the most investigated family of networks. Among other preferences its simple structure makes meshes interesting for theory and practice.

4. For meshes the problem of efficient communication between processors has been studied intensively in the last years. One of the best studied problem is the k-k routing problem. In the last ten years, many variants of this problem were solved efficiently, but some aspects of the problem remain unsolved. One of these unsolved problems is the problem of designing an oblivious routing algorithm that solve the k-k routing problem with a small constant buffer size and in a number of steps close to the best known lower bound.

   In an oblivious routing algorithm the path of a packet through the network depends only on its source and destination and hence is independent of the path of other packets. This characteristic is interesting, since it allows the design of simple and hence practical routing algorithms.

   This thesis offers a substantial contribution to solving the problem of oblivious k-k routing. An oblivious k-k routing algorithm for meshes is presented that solves the problem in an asymptotically optimal number of steps with buffer size $O(k)$. Furthermore, the proposed routing

algorithm can also be applied to OTIS and other networks and solve the problem in an asymptotically optimal number of steps and $O(k)$ buffer size for a large class of these networks.

5. Lower bounds for the number of steps needed to solve the k-k routing problem on a network are given by the diameter and bisection width of a network.

6. In the computing community there is a growing interest in optics. Optical interconnections provide high interconnections and large bandwidth. However, electronic interconnections have advantages, too. They perform better for small distances. OTIS networks (Optical Transpose Interconnection System) are networks in which optical and electronical links are used.

   In this work the diameter and upper bounds for the bisection width of OTIS networks are determined and lower bounds for the number of steps needed to solve the k-k routing problem are given. Based on this results, a new class of networks, called Extended OTIS networks, is introduced, which have smaller diameter than OTIS networks.

7. The k-k sorting problem is a data transport problem very similar to the k-k routing problem. Algorithms that solve the k-k sorting problem can be used to solve the k-k routing problem. An algorithm for the OTIS-Mesh is given that solves the k-k sorting problem with buffer size $O(k)$ in a number of steps that comes close to the diameter and bisection lower bound.

Zusammenfassung

Für die Leistungfähigkeit von Parallelrechnern, die über ein Verbindungsnetzwerk kommunizieren, ist ein effizienter Datentransport entscheidend. Ein grundlegendes Transportproblem in einem solchen Rechner ist das k-k Routing Problem. In dieser Arbeit werden Aspekte dieses Problems in r-dimensionalen Gittern und OTIS-G Netzwerken untersucht. Es wird der erste vergessliche (oblivious) Routingalgorithmus vorgestellt, der das k-k Routing Problem in diesen Netzwerken in einer asymptotisch optimalen Laufzeit bei konstanter Puffergröße löst. Für OTIS-G Netzwerke werden untere Laufzeitschranken für das untersuchte Problem angegeben, die auf dem Durchmesser und der Bisektionsweite der Netzwerke basieren. Weiterhin wird ein Algorithmus vorgestellt, der das k-k Sorting Problem mit einer Laufzeit löst, die nahe an der Bisektions- und Durchmesserschranke liegt. Basierend auf den OTIS-G Netzwerken, wird eine neue Klasse von Netzwerken eingeführt, die sogenannten Extended OTIS-G Netzwerke, die sich durch einen kleineren Durchmesser von OTIS-G Netzwerken unterscheiden.

Erklärung.

Ich versichere, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel und Quellen angefertigt habe.

Bei der Auswahl und Auswertung des Materials hat mir niemand entgeltlich oder unentgeltlich geholfen.

Weitere Personen waren an der inhaltlichen Erstellung der vorliegenden Arbeit nicht beteiligt. Insbesondere habe ich hierfür nicht die entgeltliche Hilfe von Vermittlungs- bzw. Beratungsdiensten (Promotionsberater oder anderer Personen) in Anspruch genommen. Niemand hat von mir unmittelbar oder mittelbar geldwerte Leistungen für Arbeiten erhalten, die im Zusammenhang mit dem Inhalt der vorgelegten Dissertation stehen.

Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder ähnlicher Form einer Prüfungsbehörde vorgelegt.

Ich bin darauf hingewiesen worden, dass die Unrichtigkeit der vorstehenden Erklärung als Täuschungsversuch angesehen wird und den erfolglosen Abbruch des Promotionsverfahrens zur Folge hat.

(Ort, Datum) (Unterschrift)