COMBINATORICA
Bolyai Society – Springer-Verlag

COMBINATORICA **21** (1) (2001) 95–138

DETERMINISTIC ROUTING WITH BOUNDED BUFFERS: TURNING OFFLINE INTO ONLINE PROTOCOLS*

FRIEDHELM MEYER AUF DER HEIDE, CHRISTIAN SCHEIDELER†

Received July 18, 1997

In this paper we present a deterministic protocol for routing arbitrary permutations in arbitrary networks. The protocol is analyzed in terms of the size of the network and the routing number of the network. Given a network H of n nodes, the *routing number* of H is defined as the maximum over all permutations π on $\{1,\ldots,n\}$ of the minimal number of steps to route π offline in H. We show that for any network H of size n with routing number R our protocol needs $O(\log_R n \cdot R)$ time to route any permutation in H using only constant size edge buffers. This significantly improves all previously known results on deterministic routing. In particular, our result yields optimal deterministic routing protocols for arbitrary networks with diameter $\Omega(n^\epsilon)$ or bisection width $O(n^{1-\epsilon})$, $\epsilon > 0$ constant. Furthermore we can extend our result to deterministic compact routing. This yields, e.g., a deterministic routing protocol with runtime $O(R \log n)$ for arbitrary bounded degree networks if only $O(\log n)$ bits are available at each node for storing routing information.

Our protocol is a combination of a generalized "routing via simulation" technique with an new deterministic protocol for routing h-relations in an extended version of a multibutterfly network. This protocol improves upon all previous routing protocols known for variants of the multibutterfly network. The "routing via simulation" technique used here extends a method previously introduced by the authors for designing compact routing protocols.

Mathematics Subject Classification (1991): 68W10, 68W15, 68W20

* A preliminary version appeared at FOCS '96. This work was supported in part by DFG-Sonderforschungsbereich 376 "Massive Parallelität: Algorithmen, Entwurfsmethoden, Anwendungen", by DFG Leibniz Grant Me872/6-1 and by EU ESPRIT Long Term Research Project 20244 (ALCOM-IT).

† This work was done while the author was affiliated with the Department of Mathematics and Computer Science and the Heinz Nixdorf Institute of the University of Paderborn.

1. Introduction

A fundamental problem in any parallel or distributed system is the efficient communication of data between processors. While it is possible to design a specific routing algorithm for each possible interconnection network, a much more general approach is to create a single *universal* routing algorithm that can be used in any network [23,9]. In addition to providing a unified approach to routing in standard networks, universal routing algorithms are ideally suited to routing in irregular networks that are used in wide-area networks and that arise when standard networks develop faults.

Whereas several randomized online protocols for arbitrary networks are already known under the condition that enough space for storing routing information in the nodes and packets is available (see, e.g., [21,23,29]), little is known about how space-efficiency (see, e.g., [31,26]) or bounded buffers influence the runtime. Furthermore, it is a long standing open question for which networks deterministic online protocols can compete with their randomized counterparts.

In this paper we present a deterministic online protocol that even reaches offline performance for networks with diameter $\Omega(n^\epsilon)$ or bisection width $O(n^{1-\epsilon})$, $\epsilon > 0$ constant. Moreover, this protocol can be adapted to compact routing, i.e. the situation that there is only a restricted amount of storage per processor. Our proofs use a deterministic protocol for routing arbitrary $r \cdot s$-relations in r-replicated s-ary multibutterflies in optimal time $O(\log_s n)$, and employ a technique called "routing via simulation" to use this protocol for deterministic routing in arbitrary networks.

We model an interconnection network as an undirected graph $H = (V, E)$ where $V = [n] (= \{1, \ldots, n\})$ is the set of nodes and each edge in E consists of two *links*, one in each direction. Each link contains a buffer that is able to store packets. The maximum number of packets that fit in any such buffer is called the *buffer size* of H. The nodes are working according to the *multi-port* model, that is, in each time step a vertex can send out at most one packet along each of its outgoing links. We further assume the nodes of H to work synchronously.

A *packet* consists of its *source* and *destination*, additional *routing information*, and a *message*. The source and destination need $\log n$ bits, each. Throughout this paper we restrict the space for storing additional routing information to be very small, namely of length at most $O(\log n)$. We assume the messages to have uniform length.

In order to know along which outgoing link to send a packet with a given destination, each processor needs to have some information about the structure of the network. This information is called *routing information*. The

goal of efficient compact permutation routing is to design routing protocols that route any permutation fast, using small buffers and only a few bits of routing information per node and per packet.

1.1. A model for measuring routing performance

In order to compare the performance of our online routing protocols with that of offline protocols, we use the *routing number* of a network (see, e.g., [2,28]). This parameter is defined as follows:

Let S_n be the set of all permutations on n items. Given a network H with n nodes and a permutation $\pi \in S_n$, let $R(H,\pi)$ be the minimum possible number of steps required to route packets offline in H according to π (using the multi-port model with unbounded buffers). Then the *routing number* $R(H)$ of H is defined by

$$R(H) = \max_{\pi \in S_n} R(H,\pi) \ .$$

In case that there is no risk of confusion about the graph H we will write R instead of $R(H)$.

For any fixed permutation $\pi \in S_n$, routing according to π is necessarily executed by sending the packets along paths in H. The maximum length of these paths is the *dilation* of π, the maximum number of these paths sharing an edge of H is the *congestion* of π. Clearly, if π can be routed in time at most R, there must exist a collection of paths for π with dilation and congestion at most R. This leads us to the following remark.

Remark 1.1. Let H be a network of size n with routing number R. Then, for every permutation $\pi \in S_n$, there is a path collection consisting of paths from node i to node $\pi(i)$, $1 \leq i \leq n$, which has dilation and congestion at most R.

1.2. The compact routing model

The theory of *compact routing* deals with questions about how a limited space at the processors and packets for storing routing information influences the routing performance of networks. This is mainly done by investigating, how space bounds at the processors and packets influence the realization of efficient path systems.

Previously, (apart from very few exceptions) only compact routing models have been used that establish a relationship between the available space

for storing routing information in the nodes and the *stretch factor* or the dilation of path systems obeying these space constraints. The stretch factor of a path selection scheme is defined as the maximum ratio over all pairs of nodes between the length of a route produced by the scheme and the length of a shortest path between these nodes. It was introduced by Peleg and Upfal in [31] and since then has been used in a large body of papers (see, e.g., [12,13,4,5,10,11]). The best upper bound so far has been presented by Awerbuch and Peleg in [5]. They show that, given a graph G of size n with diameter D, for every $k \geq 1$ there exists a path system with stretch factor $16k^2$ that requires $O(k \cdot n^{1/k} \log n \cdot \log D)$ bits per node and $O(\log n)$ bits per packet for storing routing information. The best lower bounds can be found in [31] and [11]. The dilation of compact path selection schemes has been studied, e.g., in [17,35,14].

The drawback of these results is that they do not consider the congestion of a path system contructed by a scheme and therefore cannot be used to analyze the routing time of protocols using these path systems. So one improvement of these models could be to not only consider the dilation, but also the congestion of a path system. We even go further by directly studying the implications of space restrictions on the *time* for routing arbitrary permutations in arbitrary networks.

1.3. Previous results

In terms of universal store-and-forward routing, many investigations have been revolving around the problem of routing packets along an arbitrary fixed path collection with congestion C and dilation D. The best time bounds within this model were obtained by Leighton, Maggs and Rao [24]. They present an optimal $O(C + D)$ time *offline* algorithm for routing along arbitrary simple path collections with congestion C and dilation D that uses constant size link buffers. Meyer auf der Heide and Vöcking [29] found an online protocol that can route n packets along an arbitrary shortcut-free path collection with congestion C and dilation D in time $O(C+D+\log n)$, w.h.p.. They further showed how to use this protocol to design optimal randomized protocols for arbitrary node-symmetric networks. Ostrovski and Rabani [30] found a randomized protocol for arbitrary simple path collections that runs in time $O(C+D+\log^{1+\epsilon} n)$, w.h.p., for any constant $\epsilon > 0$, using buffers of size C. In [9], Cypher *et al.* present a randomized protocol for arbitrary simple

Throughout the paper, the terms "*with high probability*" and "*w.h.p.*" mean "with probability at least $1 - n^{-\alpha}$" where $\alpha > 0$ is an arbitrary constant.

path collections that runs in time $O((D \log\log n + C + \frac{\log n \cdot \log\log n}{\log\log(C \cdot D)}) \cdot \frac{\log(C \cdot D)}{\log\log(C \cdot D)})$, w.h.p., and requires buffers of size $O(\frac{\log(C \cdot D)}{\log\log(C \cdot D)})$.

Not many strategies for deterministic routing are known so far. In the case of oblivious deterministic routing it is well-known that, for any network of size n and degree d, there are permutations that require at least $\Omega(\sqrt{\frac{n}{d}})$ routing steps [6,15]. Furthermore, Krizanc showed that for any constant degree network of size n and a constant number of buffers per link any deterministic source-oblivious strategy realizing all partial permutations requires $\Omega(n)$ time [18]. In case of adaptive routing, we know that any deterministic sorting algorithm on a network can also be used for deterministic permutation routing. There are, e.g., routing algorithms based on AKS sorting [1] that can be implemented on a constant degree network with runtime $\Theta(\log n)$ [20]. Furthermore, the sorting result in [8] implies that any permutation can be routed deterministically on a hypercube, shuffle-exchange, cube-connected-cyles and butterfly of size n in time $O((\log\log n)^2 \log n)$. Simpler schemes with routing time $O(\log_d n)$ are known for d-ary multibutterflies [7]. A detailed survey of further references can be found in [32] Meyer auf der Heide and Scheideler [27] were the first to present deterministic compact routing protocols. They showed that for any bounded degree node-symmetric network with diameter D there is a deterministic protocol for routing any permutation in time $O(D \log n)$ with constant size edge buffers if $O(D \log\log D)$ space is available in the nodes and $O(\log D)$ space is available in the packets for storing routing information.

1.4. New results

The main result of this paper is the design and analysis of a deterministic compact routing protocol that routes arbitrary permutations in arbitrary networks in a time close to the routing number.

Main Theorem. *Let H be an arbitrary network with n nodes, maximal degree d and routing number R. Then, for every $s \in \{2, \ldots, R\}$, there is a deterministic online protocol that routes any permutation in time $O(\log_s n \cdot R)$, if constant size buffers are available at each edge for storing packets, $\Theta((s + \log R)(d \log d + \log s) + \log n)$ space is available at each node and $\Theta(\log(s \cdot R) + \log\log n)$ space is available at each packet for storing routing information.*

The Main Theorem has several implications that are described in the following.

- If $R = \Omega(n^\epsilon)$ for some constant $\epsilon > 0$ (this is the case for all graphs with diameter $\Omega(n^\epsilon)$ or bisection width $O(n^{1-\epsilon})$), then it is even possible to route deterministically in optimal worst case time (and average time, see [28]). Such optimal protocols were previously only known for special networks like d-dimensional meshes and tori [19].
- According to [34] every n-vertex graph of genus g and maximal degree d has bisection width $O(\sqrt{gdn})$. Therefore, the Main Theorem yields asymptotically optimal deterministic routing protocols for all networks with $g \cdot d = O(n^{1-\epsilon})$, $\epsilon > 0$ constant (this includes all planar networks with degree $O(n^{1-\epsilon})$).
- For every bounded degree node-symmetric network (many standard networks and the best expanders that have an explicit construction belong to this class [26]) of size n and diameter D, the Main Theorem yields a deterministic protocol with runtime $O(\log_s n \cdot D)$ for every $s \in \{2, \ldots, D\}$. The previous best protocol for compact routing in node-symmetric networks is randomized, has runtime $O(\log_s n \cdot D)$, w.h.p., and requires $O(s \cdot D)$ space in the nodes for storing routing information [26]. Here, instead, we only need $O((s + \log D) \log s + \log n)$ space.
- For any bounded degree network of size n with routing number R, $O(\log n)$ space suffices in the nodes to route any permutation in time $O(R \log n)$. If $R = O(2^{\log n / \log\log n})$ the time bound can be improved to $O(R \frac{\log n}{\log\log n})$. Note that the space cannot be reduced below $\log n$ per node, since every node at least has to store its own number.

Our approach to achieve the relationship between routing time and the routing number of a network is an extension of the "routing via simulation" technique, introduced in [26]:

Consider networks $G = (W, F)$ and $H = (V, E)$ with $c := \frac{|V|}{|W|} \geq 1$. (In [26] only $c = 1$ is considered.) Partition H into clusters of size c such that each cluster represents a node in G and each pair of nodes within one cluster has a distance of at most $O(c)$. Furthermore let $\mathcal{P}_F$ be a path collection in H which contains paths $p(C_1, C_2)$ with endpoints in the clusters C_1 and C_2 in H only for pairs $\{u, v\} \in F$ for which C_1 simulates u and C_2 simulates v. Our strategy to simulate routing in G by H then works as follows:

Suppose, a packet with origin u and destination v travels along the path $p_G(u, v)$ in G. In order to simulate the traversal of an edge $\{x, y\} \in F$, it chooses the path $p(C_1, C_2)$ with C_1 simulating x and C_2 simulating y.

As guest graph G we use an extension of the s-ary multibutterfly network, called the *extended r-replicated s-ary multibutterfly*. The s-ary multibutterfly network was introduced in [7]. For our generalized network we show the following result.

Theorem 1.2. *Given an extended r-replicated s-ary multibutterfly of size n with $r \geq 1$ and $s \geq 4$, $r \cdot s \cdot n$ packets, $r \cdot s$ per processor, can be routed deterministically according to some arbitrary $r \cdot s$-relation in time $O(\log_s n)$.*

In [7] this time bound is only achieved for routing permutations. As we will see, the extended multibutterfly can also be used for compact routing, since its hierarchical structure allows very space-efficient routing structures, so that the space necessary for storing $\mathcal{P}_F$ dominates the space requirements in the nodes of H.

1.5. Organization of the paper

In Section 2 a new relation routing protocol for extended r-replicated s-ary multibutterfly networks is described and analyzed. This is the most involved part of the paper. The "routing via simulation" technique will be presented in detail in Section 3. Furthermore, a (non-compact) deterministic routing protocol will be constructed using this technique. In Section 4 we show how to transform this protocol into a compact deterministic protocol for arbitrary networks. Section 5 contains the conclusions and some open problems.

2. Routing in the s-ary multibutterfly

In this section we prove Theorem 1.2. Before we present the protocol we first describe, how Borodin *et al.* [7] defined their s-ary multibutterfly, and what kind of s-ary multibutterfly we need.

2.1. The s-ary multibutterfly by Borodin *et al.*

In this section we describe the s-ary multibutterfly as it is defined in [7]. The basic building block of their s-ary multibutterfly is an *s-ary m-splitter*.

The s-ary m-splitter (or (s,m)-splitter) is a bipartite graph of degree $\frac{s}{2}$ with m input nodes and m output nodes. In this graph the output nodes are partitioned into $\sqrt{s}$ output sets, each with $m/\sqrt{s}$ nodes. Every input node has $\sqrt{s}/2$ edges to each of the $\sqrt{s}$ output sets. The edges connecting the input set to each of the output sets define an expander graph with properties described in [7].

The (elementary) *s-ary d-dimensional multibutterfly* (s,d)-MBF has $d+1$ levels. The vertices at level $0 \leq i \leq d-1$ are partitioned into $\sqrt{s}^i$ sets of $m_i = \sqrt{s}^{d-i}$ consecutive nodes. Each of these sets in level i is an input set

of an s-ary m_i-splitter. The output sets of that splitter are $\sqrt{s}$ sets of size m_{i+1} in level $i+1$. Thus each node in the (s,d)-MBF is the endpoint of at most $2 \cdot \frac{s}{2} = s$ edges. Figure 1. shows the structure of a $(16,d)$-MBF.

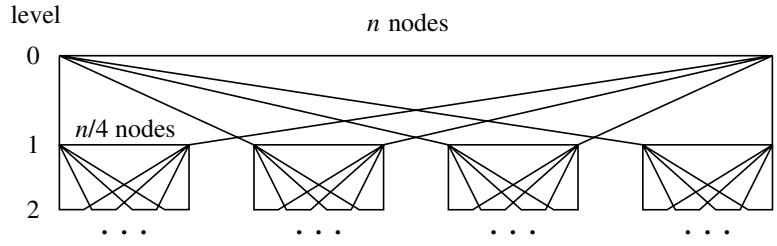

Fig. 1. The structure of a $(16,d)$-MBF.

Using the (s,d)-MBF, the following result can be shown. Its proof can be found in [7].

Theorem 2.1. *For sufficiently large s, the s-ary multibutterfly of size n can route any permutation from the inputs to the outputs in time $O(\log_s n)$.*

Using techniques in [36], Borodin *et al* also showed how to extend the s-ary multibutterfly to route any global permutation in $O(\log_s n)$ steps.

In the following, we further extend the definition of the s-ary multibutterfly such that it can be used to route any s-relation in optimal time.

2.2. The r-replicated s-ary multibutterfly

The basic building block of our (1-replicated) s-ary multibutterfly is an *s-ary m-router*.

The s-ary m-router (or (s,m)-router) is a bipartite graph with m input nodes and m output nodes. It is a combination of the following two graphs.

The first graph is called *s-ary m-distributor*. It is a directed graph with all input nodes as node set. Each input node is the starting point of s edges numbered from 1 to s. We require the edges to be chosen such that for all $i \in \{1,\ldots,s\}$ the set of all endpoints of the ith edges forms a permutation, and specific expansion properties described later are fulfilled. This graph will be used to balance the distribution of the packets in the input nodes.

The second graph we need is the *s-ary m-splitter*. As described above, in this graph the output nodes are separated into $\sqrt{s}$ output sets, each with $m/\sqrt{s}$ nodes. Every input node has $\sqrt{s}/2$ edges to each of the $\sqrt{s}$ output

sets. The edges connecting the input set to each of the output sets define an expander graph with properties we will describe later. This graph will be used to forward the packets to their destinations.

The *s-ary d-dimensional multibutterfly* (s,d)-MBF has $d+1$ levels. The nodes at level $0 \le i \le d-1$ are partitioned into $\sqrt{s}^i$ sets of $m_i = \sqrt{s}^{d-i}$ consecutive nodes. Each of these sets in level i is an input set of an s-ary m_i-router. The output sets of that router are $\sqrt{s}$ sets of size m_{i+1} in level $i{+}1$. Thus each node in our (s,d)-MBF is the endpoint of at most $2(s{+}\frac{s}{2}){=}3s$ edges.

For our compact routing protocol we also need the notion of an (s,d,k)-MBF. This multibutterfly is defined as follows. Let the (s,m,k)-router be a graph with $k \cdot m/\sqrt{s}$ input nodes and k sets of $m/\sqrt{s}$ output nodes. The input nodes are connected by an s-ary $k{\cdot}m/\sqrt{s}$-distributor. The s-ary splitter of such a router is modified in a way that every input node has $\frac{s}{2k}$ edges to each of the k output sets. We call such a splitter (s,m,k)-splitter. For $k \in \{1,\ldots,\sqrt{s}\}$, the (s,d,k)-MBF can be derived from an (s,d)-MBF by replacing the (s,m)-router at level 0 of the (s,d)-MBF by an (s,m,k)-router as shown in Figure 2.

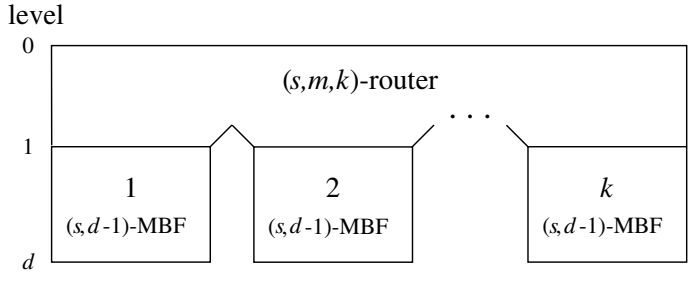

Fig. 2. The structure of an (s,d,k)-MBF.

The r-replicated s-ary multibutterfly (r,s,d,k)-MBF is defined by taking r copies of the (s,d,k)-MBF and identifying the corresponding nodes. That is, each edge now is able to forward r packets in one time step. As we will see, this multibutterfly can be used to route any $r \cdot s$-relation from the top level to the bottom level in optimal time. We will later describe how to construct an *extended* (r,s,d,k)-MBF that can even route any global $r \cdot s$-relation in optimal time.

For the rest of this section we will now present a proof of Theorem 1.2. Let n denote the number of nodes in one level of a given r-replicated s-ary

MBF. Consider first the simpler problem of routing $rs \cdot n$ packets, rs per node at the top level, to the bottom level of the MBF according to some arbitrary rs-relation. In order to do this time-efficiently we use the following protocol.

2.3. Description of the simple protocol

We partition the $rs \cdot n$ messages into 522 batches such that no more than $rs \cdot m/522$ messages from each batch traverse any m-router. Batch j has all packets with destinations in the set $\{z \mid z = j - 1 \pmod{522}\}$. For the purpose of analysis we assume that all batch j packets have been routed before transmitting batch $j+1$ packets, and we now concentrate on the routing of one batch.

Nodes at even levels of the MBF transmit in odd phases, nodes at odd levels transmit in even phases (that is, level 0 is active in phase $1, 3, \ldots$). A phase has three subphases. Consider an (s, m, k)-router in which the input nodes want to transmit. We assume that no node has more than $r \cdot s$ packets.

The task of the *balancing phase* is to distribute the packets in such a way among the input nodes that there are only very few nodes with more than $\frac{r \cdot s}{2}$ packets. As we will see, this can be obtained if each input node sends out its $p \leq r \cdot s$ packets along the edges of the m-distributor with numbers 1 to $\lceil \frac{p}{r} \rceil$ in an arbitrary order. Note that, according to the definition of the m-distributor, each input node receives at most $r \cdot s$ packets.

The task of the *placement phase* is to distribute the packets in such a way among the input nodes that there are only very few nodes having more than $\frac{r \cdot s}{4k}$ packets that have to be sent to the same output set. As we will see, this can be obtained if the packets are distributed among the s edges of the m-distributor in a specific way, before sending them out.

The *delivery phase* consists of three steps. Its task is to send as many packets as possible to output sets prescribed by their destinations. For each input node, only the first $\frac{r \cdot s}{4k}$ packets to each of the k output sets are declared active. In the first step each input node sends for each output set to which it has messages to transmit a request message along all $\frac{s}{2k}$ edges of the m-splitter to that output set. An output node that currently stores at most $\frac{rs}{2}$ packets replies in the second step with a ready message to all input nodes that sent a request to it. In the third step each input node sends up to r active packets to each node that sent it a ready message.

2.4. Analysis of the simple protocol

We first analyze the routing of one batch. Consider an (s,m,k)-router in a phase in which the inputs of that router are active. Let x (resp. x') be the total number of packets that are stored in the input nodes of that splitter at the beginning (resp. end) of that phase. Let y be the number of packets stored in output nodes of the splitter at the beginning of this phase.

Lemma 2.2.

$$x' \leq \frac{400 \log s}{\sqrt{s}}(x+y) \ .$$

Proof. Both the balancing phase and the placement phase require the packets to be distributed among the edges of the m-distributor in a suitable way. For each input node v, this will be done with the help of an assignment graph that is defined as follows.

Assume that every packet has one out of c possible colors. The *assignment graph* $A=(P_A,D_A,E_A)$ is defined as a bipartite undirected graph with node sets P_A and D_A, and a set of edges E_A. $P_A=\{v_{i,j} \mid i\in\{1,\ldots,c\}, j\in\{1,\ldots,s\}\}$ consists of $c\cdot s$ nodes, s nodes for each color, and D_A consists of s nodes representing all s edges leaving v in the m-distributor. Each node in P_A has d edges to nodes in D_A. Consider v to have p_i packets with color i, partitioned into $\lceil \frac{p_i}{r} \rceil$ *blocks* of size at most r. We allow the jth of these blocks to be assigned only to nodes in the set $\Gamma(\{v_{i,j}\})$. (For any node set U, $\Gamma(U)$ is defined as the set of all nodes v that are adjacent to a node $u\in U$.) In case that under this restriction every block can be assigned to a different node in D_A, each block is sent along that edge of the m-distributor that is represented by the node in D_A that has been assigned to the block. If every input node uses the same assignment graph then, as will be shown in Proposition 2.8, the edges of the m-distributor can be distributed in such a way among the input nodes that, after sending the blocks of packets along the edges assigned to them, the distribution of the packets among the inputs is close to be balanced w.r.t any color.

For the balancing phase an assignment graph can easily be constructed. The task of this phase is to distribute the packets among the input nodes in such a way that there is only a very small portion of packets left that is stored in nodes with more than $\frac{r\cdot s}{2}$ packets. Hence we need only one color. Consider input v to have p packets, partitioned into $\lceil \frac{p}{r} \rceil$ blocks of size at most r. Then we simply assign the jth edge of the m-distributor to the jth of these blocks. The underlying assignment graph has degree $d=1$. Consider marking the first (at most) $\frac{r\cdot s}{2}$ packets in each input node after the balancing

phase. Then we are interested in the total number of unmarked packets. An upper bound for this number will be given in the following proposition.

Proposition 2.3. *There exists an s-ary m-distributor such that after the balancing phase at most $\frac{48x\log s}{s}$ unmarked packets are stored in the input nodes.*

Proof. The result follows from Proposition 2.8 with $d=1$, $c=1$, $z=\frac{s}{2}$ and $\epsilon \leq \frac{1}{522} \leq \frac{c\cdot z}{12ed\cdot s}$. ∎

For the placement phase it is more difficult to find a suitable assignment graph. The task of the placement phase is to distribute the packets in such a way among the input nodes that there are only very few nodes having more than $\frac{r\cdot s}{4k}$ packets that have to be sent to the same output set. Hence we need k colors representing the k output sets of the (s,m,k)-router. If we only concentrate on assigning edges of the m-distributor to marked packets, we can prove the following result.

Proposition 2.4. *There is an assignment graph A of degree 4 such that, for any choice of colors for the at most $\frac{r\cdot s}{2}$ marked packets stored in an input node, A can be used to assign at most one block of marked packets to each edge such that the j-th block of color $i\in\{1,\ldots,k\}$ is assigned to an edge in $\Gamma(\{v_{i,j}\})$.*

Proof. Consider a fixed input node v. The proof consists of a probabilistic argument. In particular, we show that for randomly chosen endpoints in D_A for the edges in E_A and suitably chosen d the probability that the resulting graph A does not fulfill the proposition is smaller than one. Therefore, there exists an assignment graph A such that the marked packets can be distributed among the edges in such a way that each edge gets at most one block of packets.

Let $p\leq k$ denote the number of colors used by the marked packets in v, let i_j be the number of blocks of marked packets in v assigned to the jth of these p colors. Since each input node stores at most $\frac{r\cdot s}{2}$ marked packets, the number q of blocks is at most $\sum_{j=1}^{p} i_j \leq \frac{s}{2}+p$ (for each of the p colors there may exist a block with less than r packets). There are at most $\binom{s}{q-1}$ possibilities for choosing a subset of $q-1$ out of s possible nodes in D_A. The probability that all alternatives for the blocks point to nodes within such a subset is bounded by $(\frac{q-1}{s})^{d\cdot q}$. Then the probability that for any choice of colors for the packets there is a subset S of nodes in P_A representing blocks

of packets with $\Gamma(S)<|S|$ is bounded above by

$$\sum_{p=1}^{k}\binom{k}{p}\sum_{\substack{1\le i_1,\dots,i_p\le s,\\ \sum_j i_j\le s/2+p}}\binom{s}{\sum_j i_j-1}\left(\frac{\sum_j i_j-1}{s}\right)^{d\sum_j i_j}$$

$$\le\sum_{p=1}^{k}\binom{k}{p}\sum_{q=p}^{s/2+p}\binom{q-1}{p-1}\binom{s}{q-1}\left(\frac{q-1}{s}\right)^{d\cdot q} \tag{1}$$

In order to simplify the formula, we need the following claim.

Claim 2.5. *For a sufficiently large s and $d\ge 4$,*

$$\binom{s}{p}\left(\frac{p}{s}\right)^{d\cdot p}\ge\binom{s}{q-1}\left(\frac{q-1}{s}\right)^{d\cdot q}$$

for all $p\in\{1,\dots,k\}$ and $q\in\{p,\dots,s/2+p\}$.

Proof. It holds

$$\binom{s}{q}\left(\frac{q}{s}\right)^{d\cdot q}\ge\binom{s}{q+1}\left(\frac{q+1}{s}\right)^{d(q+1)}$$

$$\Leftrightarrow\quad\frac{\binom{s}{q}}{\binom{s}{q+1}}\ge\left(1+\frac{1}{q}\right)^{d\cdot q}\left(\frac{q+1}{s}\right)^{d}$$

$$\Leftarrow\quad\frac{q+1}{s-q}\ge\left(\frac{\mathrm{e}(q+1)}{s}\right)^{d},$$

which is true for all $d\ge 4$ and $q\in\{p,\dots,\frac{s}{2\mathrm{e}}-1\}$, if s is sufficiently large. Hence $\binom{s}{p}(\frac{p}{s})^{d\cdot p}\ge\binom{s}{q-1}(\frac{q-1}{s})^{d\cdot q}$ for all $q\in\{p+1,\dots,\frac{s}{2\mathrm{e}}-1\}$. In case that $q\in\{\frac{s}{2\mathrm{e}},\dots,\frac{s}{2}+p\}$ and s sufficiently large (recall that $p\le\sqrt{s}$), we get

$$\binom{s}{q-1}\left(\frac{q-1}{s}\right)^{d\cdot q}\le\mathrm{e}^q\left(\frac{q-1}{s}\right)^{(d-1)q}\le\left(\frac{1}{2}\right)^{q}\le$$

$$\left(\frac{p}{s}\right)^{(d-1)p}\le\binom{s}{p}\left(\frac{p}{s}\right)^{d\cdot p}.$$

In case that $q=p$ and $p>1$ (the case $p=1$ is obviously correct) we get

$$\binom{s}{p}\left(\frac{p}{s}\right)^{d\cdot p}\ge\binom{s}{p-1}\left(\frac{p-1}{s}\right)^{d\cdot p}$$

$$\Leftrightarrow\quad\left(\frac{p}{p-1}\right)^{d\cdot p}\ge\frac{p}{s-p+1},$$

which is also true. Hence the claim follows. ∎

Let $d \geq 4$ and s be sufficiently large (note that $k \leq \sqrt{s}$). Then it follows from Claim 2.5 that

$$
\begin{aligned}
(1) &\leq \sum_{p=1}^{k} \binom{k}{p} \binom{s}{p} \left(\frac{p}{s}\right)^{d \cdot p} \sum_{q=p}^{s/2+p} \binom{q-1}{p-1} \\
&\leq \sum_{p=1}^{k} \binom{k}{p} \left(\frac{\mathrm{e}s}{p}\right)^{p} \left(\frac{p}{s}\right)^{d \cdot p} \binom{s/2+p}{p} \\
&\leq \sum_{p=1}^{k} \binom{k}{p} \left(\frac{\mathrm{e}(s/2+p)}{p}\right)^{p} \mathrm{e}^{p} \left(\frac{p}{s}\right)^{(d-1)p} \\
&\leq \sum_{p=1}^{k} \left(\frac{\mathrm{e}k}{p}\right)^{p} \mathrm{e}^{2p} \left(\frac{p}{s}\right)^{(d-2)p} \\
&\leq \sum_{p=1}^{k} \left(\frac{p}{s}\right)^{(d-3)p} \quad < \quad 1 \ .
\end{aligned}
$$

The following result can be shown about matchings in bipartite graphs. Its proof is similar to the proof of the well-known Matching Theorem by Hall (see [21], p. 191).

Theorem 2.6. *Let $G = (V_1, V_2, E)$ be an arbitrary bipartite graph with $|V_1| \leq |V_2|$. Then G contains a matching of size $|V_1|$ if for any subset $S \subseteq V_1$ it holds that $|\Gamma(S)| \geq |S|$.*

Using this theorem it follows that the probability is smaller than one that, for randomly chosen endpoints in D_A for the edges in E_A, there exists a coloring of the marked packets such that an assignment of marked packets to nodes in D_A as described above is not possible. Thus the proposition holds. ∎

Let each edge of the distributor represent r channels. Then the unmarked packets will be assigned to those of the $r \cdot s$ channels that are not used by the marked packets. Although we do not consider the unmarked packets in the following, it is important to let them participate in the placement phase to ensure that after the placement phase every input node has at most $r \cdot s$ packets. Let us call the first $\frac{r \cdot s}{4k}$ marked packets for each output set that are stored in each input node after the placement phase *active*. Then we can prove the following proposition.

Proposition 2.7. *There exists an s-ary m-distributor such that at most $\frac{384x \log s}{\sqrt{s}}$ marked packets are not active at the end of the placement phase.*

Proof. The result follows from Proposition 2.8 with $d = 4$, $c = k \leq \sqrt{s}$, $z = \frac{s}{4k}$, and $\epsilon \leq \frac{1}{522} \leq \frac{c \cdot z}{12ed \cdot s}$. ∎

It remains to prove that, for a given assignment graph A, there exists an m-distributor such that the packets are close to be balanced among the input nodes w.r.t. any color.

Proposition 2.8. *Let A be an assignment graph for c colors that has degree d and can be used to assign every edge of the m-distributor to at most one block of packets of any color in every input node. Consider marking for each color the first z packets in each input node after the packets have been sent along the edges of the m-distributor. If there are at most $\frac{\epsilon rsm}{c}$ packets of each color stored in the input nodes, $z \geq \sqrt{s}/4$, $\epsilon \leq \frac{c \cdot z}{12ed \cdot s}$, and s sufficiently large, then there exists an s-ary m-distributor such that the total number of unmarked packets is at most*

$$\frac{24d \cdot x \log s}{z} .$$

Proof. We use a probabilistic proof to show that there is a suitable distribution of the edges of the m-distributor among the input nodes such that Proposition 2.8 holds.

Consider a fixed color $\gamma \in \{1, \ldots, c\}$. Let x_γ denote the number of packets stored in the input nodes that have color γ. Let $p \geq z$, $q = \frac{p}{2d}$, and b_i denote the number of input nodes that have at least i blocks of packets with color γ, $i \in \{q, \ldots, \frac{s}{d}\}$. Note that if an input node has i blocks of packets with color γ then it has at least $f(i) := (i-1)r+1$ packets with color γ. Thus the maximal number of input nodes with $i > q$ blocks given x_γ and $b_q, \ldots, b_{i-1}$ is at most

$$\min \left\{ \left\lceil \frac{x_\gamma - g(b_q, \ldots, b_{i-1})}{r} \right\rceil , \; b_{i-1} \right\}$$

with $g(b_q, \ldots, b_{i-1}) := f(q) \cdot b_q + \sum_{j=q+1}^{i-1} r \cdot b_j$. The following claim reveals why the assignment graph is so important for balancing the packets among the input nodes.

Claim 2.9. *If all input nodes use the same assignment graph A of degree d then the s edges reaching any input v can be numbered from 1 to s such that a block of packets with color γ that arrives at v via edge j implies at least $\lfloor \frac{j}{d} \rfloor$ blocks of packets of color γ in its origin before the packets have been sent out.*

Proof. Let all input nodes use the same assignment graph. Then it holds that, since for every $j \in \{1,\ldots,s\}$ the endpoints of the jth edges from all input nodes form a permutation, each input node is the endpoint of s edges representing the complete node set D_A. Consider the edges to be numbered from 1 to s such that if edge e has a lower number then edge e' then $\min\{j \mid e \in \Gamma(\{v_{\gamma,j}\})\} \leq \min\{j \mid e' \in \Gamma(\{v_{\gamma,j}\})\}$. Since the jth block of color γ is only allowed to choose among d nodes in D_A it follows that if a block of color γ reaches a node v via edge j then its origin must have had at least $\lfloor \frac{j}{d} \rfloor$ blocks of packets with color γ. ∎

With the help of this claim we can prove the following claim.

Claim 2.10. *The probability that after the placement phase at least u input nodes have at least p blocks of packets, each, with color γ is bounded by*

$$\binom{m}{u}\binom{m}{\frac{x_\gamma}{f(p/2d)}}\left(\frac{s}{d}\right)^{\frac{x_\gamma}{f(p/2d)}}\left(\frac{2ed\cdot x_\gamma}{p\cdot r\cdot m}\right)^{\frac{u\cdot p}{2}} .$$

Proof. For randomly chosen endpoints of the edges of the m-distributor, the probability that after the placement phase at least u input nodes have at least p blocks of packets, each, with color γ is bounded by

$$\binom{m}{u}\sum_{\substack{b_q,\ldots,b_{s/d}\geq 0,\\ \forall i: b_i \geq b_{i+1},\ g(b_q,\ldots,b_{s/d})\leq x_\gamma}}\left[\sum_{\frac{p}{2}\leq i_1<\ldots<i_{p/2}\leq s}\prod_{j=1}^{p/2}\frac{b_{\lfloor i_j/d\rfloor}}{m}\right]^u . \tag{2}$$

This formula is derived as follows.

- There are $\binom{m}{u}$ ways to choose a set U of input nodes of size u.
- If p blocks of packets are sent to input node v, there must exist at least $\frac{p}{2}$ blocks that use an edge set $\{i_1,\ldots,i_{p/2}\} \subseteq \{\frac{p}{2},\ldots,s\}$ with numbers chosen as defined in Claim 2.9. This entails a probability of $\frac{b_{\lfloor i_j/d\rfloor}}{m}$ that the i_jth edge is used by a block with color γ. Thus we get that, for each input node v in U, the probability that it gets at least p blocks of marked packets with color γ is at most

$$\sum_{\frac{p}{2}\leq i_1<\ldots<i_{p/2}\leq s}\prod_{j=1}^{p/2}\frac{b_{\lfloor i_j/d\rfloor}}{m} . \tag{3}$$

  Since there is only a limited number of edges that have their origin in nodes with at least q packets of color γ, the probabilities for the nodes in

U are negatively correlated and therefore can be regarded as independent for an upper bound. This means that we get an overall probability of at most $(3)^u$.

Because of $f(q)\cdot b_q+\sum_{j=q+1}^{s/d} r\cdot b_j \le x_\gamma$ (see the formula for g above) it holds that

$$\begin{aligned}\sum_{j=\frac{p}{2}}^{s} b_{\lfloor j/d\rfloor} \le d\cdot \sum_{j=\frac{p}{2d}}^{s/d} b_j \;&=\; d\left(b_q+\frac{1}{r}\sum_{j=q+1}^{s/d} r\cdot b_j\right)\\ &\le d\left(b_q+\frac{1}{r}\left(x_\gamma-f(q)b_q\right)\right) \;\le\; \frac{d\cdot x_\gamma}{r}\;.\end{aligned}$$

Thus we get

$$\begin{aligned}(3) &\le \frac{1}{(p/2)!}\sum_{i_1,\ldots,i_{p/2}\in\{\frac{p}{2},\ldots,s\}}\prod_{j=1}^{p/2}\frac{b_{\lfloor i_j/d\rfloor}}{m}\\ &= \frac{1}{(p/2)!}\sum_{i_1,\ldots,i_{p/2-1}\in\{\frac{p}{2},\ldots,s\}}\prod_{j=1}^{p/2-1}\frac{b_{\lfloor i_j/d\rfloor}}{m}\sum_{i_{p/2}\in\{\frac{p}{2},\ldots,s\}}\frac{b_{\lfloor i_j/d\rfloor}}{m}\\ &\le \frac{1}{(p/2)!}\sum_{i_1,\ldots,i_{p/2-1}\in\{\frac{p}{2},\ldots,s\}}\prod_{j=1}^{p/2-1}\frac{b_{\lfloor i_j/d\rfloor}}{m}\cdot\frac{d\cdot x_\gamma}{r\cdot m}\\ &\le \cdots \le \frac{1}{(p/2)!}\left(\frac{d\cdot x_\gamma}{r\cdot m}\right)^{p/2} \le \left(\frac{2\mathrm{e}d\cdot x_\gamma}{p\cdot r\cdot m}\right)^{p/2}.\end{aligned}$$

Using this in (2) we get that the probability that at least u input nodes have at least p blocks of marked packets, each, with color γ is bounded by

$$\begin{aligned}&\binom{m}{u}\sum_{\substack{b_q,\ldots,b_s\ge 0,\\ \forall i: b_i\ge b_{s/d},\, g(b_q,\ldots,b_{s/d})\le x_\gamma}}\left(\frac{2\mathrm{e}d\cdot x_\gamma}{p\cdot r\cdot m}\right)^{\frac{u\cdot p}{2}}\\ &\le \binom{m}{u}\binom{m}{\frac{x_\gamma}{f(q)}}\left(\frac{s}{d}\right)^{\frac{x_\gamma}{f(q)}}\left(\frac{2\mathrm{e}d\cdot x_\gamma}{p\cdot r\cdot m}\right)^{\frac{u\cdot p}{2}}.\end{aligned}$$

Since $q=\frac{p}{2d}$, the claim follows. ∎

Let $u_{x_\gamma,p}\cdot p=\frac{4x_\gamma\log s}{f(p/2d)}$ and s be sufficiently large. Then the probability that at least $u_{x_\gamma,p}$ output nodes have at least $p\ge z$ blocks of marked packets, each,

with color γ is bounded by

$$\sum_{\substack{f(p/2d)\le x_\gamma\le \frac{\epsilon r s m}{c},\\ f(p/2d)\mid x_\gamma}} \binom{m}{u_{x_\gamma,p}} \binom{m}{\frac{x_\gamma}{f(p/2d)}} \left(\frac{s}{d}\right)^{\frac{x_\gamma}{f(p/2d)}} \left(\frac{2\mathrm{e}d\cdot x_\gamma}{p\cdot r\cdot m}\right)^{\frac{p\cdot u_{x_\gamma,p}}{2}}$$

$$= \sum_{v=1}^{\frac{\epsilon r s m}{c f(p/2d)}} \binom{m}{\frac{4v\log s}{p}} \binom{m}{v} \left(\frac{s}{d}\right)^{v} \left(\frac{\mathrm{e}v}{m}\right)^{2v\log s}$$

$$\le \sum_{v=1}^{\frac{\epsilon r s m}{c f(p/2d)}} \left(\frac{\mathrm{e}p\cdot m}{4v\log s}\right)^{\frac{4v\log s}{p}} \left(\frac{\mathrm{e}\ s\ m}{d\cdot v}\right)^{v} \left(\frac{\mathrm{e}v}{m}\right)^{2v\log s}$$

$$\overset{p\ge\sqrt{s}/4}{\le} \sum_{v=1}^{\frac{\epsilon r s m}{c f(p/2d)}} \left(\frac{m}{\mathrm{e}v}\right)^{v} \left(\frac{m}{\mathrm{e}v}\right)^{v(\log s-1)} \left(\frac{\mathrm{e}v}{m}\right)^{2v\log s}$$

$$\overset{\epsilon\le\frac{c\cdot f(p/2d)}{4ers}}{\le} \sum_{v=1}^{\frac{\epsilon r s m}{c f(p/2d)}} \left(\frac{1}{4}\right)^{v\log s} \le \frac{2}{s^2}$$

Summing over all $p=z$ to s this yields a probability smaller than 1 if $s>2$. Note that for $\frac{z}{2d}\ge 3$ we have

$$\epsilon \le \frac{c\cdot z}{12\mathrm{e}d\cdot s} \le \frac{c(z/(2d)-1)}{4\mathrm{e}s} \le \frac{c\cdot f(p/2d)}{4ers} \ .$$

Since there exists an m-distributor with $u_{x_\gamma,p}\cdot p \le \frac{4x_\gamma\log s}{f(p/2d)}$ for any distribution of the packets such that the assignment graph can be applied, at most $\frac{4d\cdot x_\gamma\log s}{p\cdot f(p/2d)}$ input nodes have at least $p\ge z$ blocks of marked packets after sending them along the edges of the distributor.

Let c_i be the number of input nodes that have exactly i blocks of marked packets after sending them along the edges of the distributor, $i\in\{z,\dots,s\}$. Then the total number of unmarked blocks of packets is at most $\sum_{i=z}^{s} i\cdot c_i$. Since we require the c_i to obey

$$\sum_{i=p}^{s} c_i \le \frac{4d\cdot x_\gamma\log s}{p\cdot f(p/2d)}$$

for all $p\ge z$, $\sum_{i=z}^{s} i\cdot c_i$ gets maximal if we set

$$c_s = \frac{4d\cdot x_\gamma\log s}{s\cdot f(s/2d)} \ ,$$

$$c_{s-1} = \frac{4d\cdot x_\gamma\log s}{(s-1)\cdot f((s-1)/2d)} - \frac{4d\cdot x_\gamma\log s}{s\cdot f(s/2d)} \ ,$$

$$c_{s-2} = \frac{4d \cdot x_\gamma \log s}{(s-2) \cdot f((s-2)/2d)} - \frac{4d \cdot x_\gamma \log s}{(s-1) \cdot f((s-1)/2d)} ,$$
$$\dots$$

From this we conclude that for a sufficiently large s and $\frac{z}{2d} \geq 3$ the edges of the m-distributor can be chosen in such a way that at most

$$\begin{aligned} & r \left[\sum_{p=z}^{s-1} p \left(\frac{4x_\gamma \log s}{p \cdot f(p/2d)} - \frac{4x_\gamma \log s}{(p+1) \cdot f((p+1)/2d)} \right) + s \cdot \frac{4x_\gamma \log s}{sf(s/2d)} \right] \\ & \quad \leq r \left[\sum_{p=z}^{s-1} p \left(\frac{12d \cdot x_\gamma \log s}{r \cdot p^2} - \frac{12d \cdot x_\gamma \log s}{r(p+1)^2} \right) + s \cdot \frac{12d \cdot x_\gamma \log s}{r \cdot s^2} \right] \\ & \quad = r \left[z \cdot \frac{12d \cdot x_\gamma \log s}{r \cdot z^2} + \sum_{p=z+1}^{s} \frac{12d \cdot x_\gamma \log s}{rp^2} \right] \\ & \quad \leq \frac{24d \cdot x_\gamma \log s}{z} \end{aligned}$$

packets with color γ are not marked. Summing over all colors yields the proposition. ∎

Note that we showed above that for randomly distributed edges the m-distributor fails to fulfill the requirements of the balancing phase resp. the placement phase with probability at most $2/s$. Thus for randomly distributed edges a single m-distributor fails to fulfill the requirements of both the balancing phase *and* the placement phase with probability at most $4/s$. Hence for sufficiently large s there exists an m-distributor that can be used for both the balancing phase and the placement phase.

Next we analyze the delivery phase. Since each input node wants to send at most $\frac{s}{2k} \cdot k = \frac{s}{2}$ requests, the edges of the (s, m, k)-splitter can be distributed in such a way that each output node has degree $\frac{s}{2}$ and therefore receives at most $\frac{s}{2}$ requests. This ensures that, if an output node stores at most $\frac{rs}{2}$ packets at the beginning of the delivery phase, it can accept r new packets from every edge of the (s, m, k)-splitter without having more than $r \cdot s$ packets afterwards. Since output nodes with more than $\frac{rs}{2}$ packets may cause problems, we need a bound for the number of packets that can not be sent from the input nodes to the output nodes.

Proposition 2.11. *Let y be the total number of packets stored in the output nodes. For all $k \in \{1, \dots, \sqrt{s}\}$ there exists an (s, m, k)-splitter for distributing the requests for the active packets in such a way among the outputs that at most $\frac{4y}{s}$ active packets fail to be sent to their output sets.*

Proof. Consider a fixed output set Y. Let y' be the number of packets stored in its nodes. The proof consists of a probabilistic argument. In particular, we show that for randomly chosen edges such that each input node has $\frac{s}{2k}$ and output node has $\frac{s}{2}$ endpoints the probability is less than one that, for any $y' \leq \frac{\epsilon rsm}{k}$, there is a subset $C_{y'}$ of input nodes (of size depending on y') such that, for any choice of $\frac{s}{4k}$ edges out of the $\frac{s}{2k}$ edges leaving input nodes in $C_{y'}$, all remaining edges point to output nodes with more than $\frac{rs}{2}$ packets. Thus there exists a bipartite graph that restricts the number of active packets that fail to reach Y to be at most $\frac{r \cdot s}{4k}|C_{y'}|$.

Let m be the number of input and output nodes of the splitter and $c_{y'}$ be the size of $C_{y'}$. Then the probability described above is bounded by

$$\sum_{\substack{\frac{rs}{2} \leq y' \leq \frac{\epsilon rsm}{k}, \\ \frac{rs}{2} | y'}} \binom{m}{c_{y'}} \left(\frac{\frac{s}{2k}}{\frac{s}{4k}}\right)^{c_{y'}} \binom{m/k}{\frac{y'}{rs/2}} \left(\frac{\frac{y'}{rs/2}}{m/k}\right)^{c_{y'} \cdot \frac{s}{4k}}$$

This is because the number of ways to choose $c_{y'}$ out of m input nodes for $C_{y'}$ is $\binom{m}{c_{y'}}$, and the number of ways to choose $\frac{s}{4k}$ out of the $\frac{s}{2k}$ edges leaving input nodes in $C_{y'}$ is bounded by $\binom{s/2k}{s/4k}^{c_{y'}}$. Furthermore there are at most $\frac{y'}{rs/2}$ output nodes with more than $\frac{rs}{2}$ packets. The number of ways to choose a subset of output nodes such that all output nodes with more than $\frac{rs}{2}$ packets are included is therefore bounded by $\binom{m/k}{y'/(rs/2)}$. The probability that one chosen edge from a node in $C_{y'}$ has its endpoint in such a subset is $\frac{y'/(rs/2)}{m/k}$. Since we require every output node to have a fixed degree of $\frac{s}{2}$, the probabilities for the $c_{y'} \cdot \frac{s}{4k}$ chosen edges to fall into the same subset of $y'/(rs/2)$ nodes are negatively correlated and therefore can be regarded as independent for an upper bound. Thus the probability that all of the $\frac{s}{4k}$ edges chosen for each input in $C_{y'}$ belong to a subset of output nodes with more than $\frac{rs}{2}$ packets is at most $(\frac{y'/(rs/2)}{m/k})^{c_{y'} \cdot s/4k}$. Let $c_{y'} \cdot \frac{s}{4k} = \frac{4y'}{rs}$ and s be sufficiently large (note that $k \leq \sqrt{s}$). Then the overall probability described above is bounded by

$$\sum_{\substack{\frac{rs}{2} \leq y' \leq \frac{\epsilon rsm}{k}, \\ \frac{rs}{2} | y'}} \binom{m}{c_{y'}} \left(\frac{\frac{s}{2k}}{\frac{s}{4k}}\right)^{c_{y'}} \binom{m/k}{\frac{y'}{rs/2}} \left(\frac{\frac{y'}{rs/2}}{m/k}\right)^{c_{y'} \cdot \frac{s}{4k}}$$

$$= \sum_{z=1}^{2\epsilon m/k} \binom{m}{\frac{8k \cdot z}{s}} \left(\frac{\frac{s}{2k}}{\frac{s}{4k}}\right)^{\frac{8k \cdot z}{s}} \binom{m/k}{z} \left(\frac{z}{m/k}\right)^{2z}$$

$$
\begin{aligned}
&\leq \sum_{z=1}^{2\epsilon m/k} \left(\frac{\mathrm{e}s \cdot m}{8k \cdot z}\right)^{\frac{8k \cdot z}{s}} (2\mathrm{e})^{\frac{8k \cdot z}{s}} \left(\frac{\mathrm{e}m}{k \cdot z}\right)^{z} \left(\frac{z}{m/k}\right)^{2z} \\
&\leq \sum_{z=1}^{2\epsilon m/k} \left(\frac{2s \cdot m}{k \cdot z}\right)^{\frac{8k \cdot z}{s}} \left(\frac{\mathrm{e}k \cdot z}{m}\right)^{z} \\
&\leq \sum_{z=1}^{2\epsilon m/k} \left(\frac{m}{\mathrm{e}k \cdot z}\right)^{\frac{k \cdot z \log s}{s}} \left(\frac{\mathrm{e}k \cdot z}{m}\right)^{z} \\
&\leq \sum_{z=1}^{2\epsilon m/k} \left(\frac{\mathrm{e}k \cdot z}{m}\right)^{z/2} \overset{\epsilon \leq \frac{1}{18\mathrm{e}}}{\leq} \sum_{z=1}^{2\epsilon m/k} \left(\frac{1}{3}\right)^{z} \; < \; 1 \; .
\end{aligned}
$$

Hence there exists an (s,m,k)-splitter such that $c_{y'} \leq \frac{4k}{s} \cdot \frac{4y'}{rs} = \frac{16ky'}{rs^2}$ for all $y' \leq \frac{\epsilon rsm}{k}$. Thus at most $\frac{r \cdot s}{4k} \cdot \frac{16ky'}{rs^2} = \frac{4y'}{s}$ active packets fail to be sent to Y. Summing over all output sets yields the proposition. ∎

From Proposition 2.3, Proposition 2.7, and Proposition 2.11 it follows that the total number of packets stored in input nodes at the beginning of the phase reduces to at most

$$x' = \frac{48x \log s}{s} + \frac{384x \log s}{\sqrt{s}} + \frac{4y}{s} \; \leq \; \frac{400 \log s}{\sqrt{s}}(x+y)$$

packets at the end of the phase assuming that s is large enough. This completes the proof of Lemma 2.2. ∎

Let X_i^t denote the number of packets in level i after the execution of phase t. Then the following theorem can be shown by using a potential function as described in [22] and [36].

Theorem 2.12. *There is a deterministic multi-port scheme on an r-replicated s-ary MBF of size N that routes any $r \cdot s$-relation from the top level to the bottom level in $O(\log_s N)$ steps if s is sufficiently large.*

Proof. We analyze the progress of the routing algorithm in terms of a potential function. Let n be the number of nodes in each level of a given (s,d,k)-MBF and $w = s^{1/4}/(\log s)^{1/2}$. The *potential* of a packet after phase t is w^i if after the execution of that phase the packet is in level $d-i$ of the network. (When a packet reaches its destination its potential is 1.) Let $\Phi(t)$ denote the sum of the potentials of the n packets after phase t, that is,

$$\Phi(t) = \sum_{i=0}^{d} X_i^t w^{d-i} \; .$$

Clearly $\Phi(0) = rs \cdot n \cdot w^d$, and routing a batch terminates at the first phase τ such that $\Phi(\tau) < r$.

Let $f(s) = \frac{400 \log s}{\sqrt{s}}$. Assume that t and i are even. Then by Lemma 2.2

$$X_i^{t+1} \le f(s)(X_i^t + X_{i+1}^t) \quad \text{and} \quad X_{i+1}^{t+1} \le X_i^t + X_{i+1}^t \ ,$$

and after the next phase

$$X_i^{t+2} \le X_{i-2}^t + X_{i-1}^t + f(s)(X_i^t + X_{i+1}^t)$$

and

$$X_{i+1}^{t+2} \le f(s)(X_i^t + X_{i+1}^t + f(s)(X_{i+2}^t + X_{i+3}^t)) \ .$$

Plugging these bounds into the potential function and applying the equation

$$X_{i+j}^t w^{d-i} = w^j \cdot X_{i+j}^t w^{d-(i+j)}$$

yields

$$\begin{aligned}
\Phi(t+2) \le & \sum_{0 \le i \le d,\ i \text{ even}} (X_{i-2}^t + X_{i-1}^t + f(s)(X_i^t + X_{i+1}^t)) w^{d-i} + \\
& \sum_{0 \le i \le d,\ i \text{ odd}} f(s)(X_{i-1}^t + X_i^t + f(s)(X_{i+1}^t + X_{i+2}^t))) w^{d-i} \\
\le & \sum_{0 \le i \le d,\ i \text{ even}} \left(\frac{1}{w^2} + f(s) + \frac{f(s)}{w} + f(s)^2 w \right) X_i^t w^{d-i} + \\
& \sum_{0 \le i \le d,\ i \text{ odd}} \left(\frac{1}{w} + f(s)w + f(s) + (f(s)w)^2 \right) X_i^t w^{d-i} \\
\le & \sum_{i=0}^{d} \left(\frac{1}{w} + f(s)w + f(s) + (f(s)w)^2 \right) X_i^t w^{d-i} \ .
\end{aligned}$$

Thus the potential function is decreased by at least

$$\frac{1}{w} + f(s)w + f(s) + (f(s)w)^2 = O(w^{-1})$$

every two phases, and for $\tau = O(\log_s N)$, $\Phi(\tau) < r$. Since there are only $O(1)$ batches to route the theorem follows. ∎

We now show how to extend this scheme to route any global $r \cdot s$-relation in $O(\log_s N)$ steps. To simplify the presentation we use a topology with $3N$ nodes we simply call $G(r,s,d,k)$ for routing $r \cdot s \cdot N$ packets. $G(r,s,d,k)$ consists of $3d$ levels of $n = k\sqrt{s}^{d-1}$ nodes each and uses edges that can forward r packets in one step. The first d levels are connected by $(s,n,1)$-routers, the second d levels represent the (r,s,d,k)-MBF, and the last d levels are connected to each other by $\frac{s}{2}$ forward edges between any input i and output i for all $i \in \{1,\ldots,n\}$.

Overlapping these three stages and identifying the corresponding nodes yields an N-node topology of depth d with degree $2(2s+s)+2\cdot\frac{s}{2}=7s$, that can simulate one step in $G(r,s,d,k)$ with constant delay. Such a network is called *extended s-ary multibutterfly* and denoted by (r,s,d,k)-XBF.

Initially, all the $r \cdot s \cdot N$ packets reside in the first d levels of $G(r,s,d,k)$. All the final destinations of the packets are in the nodes of the last d levels. Clearly, there is a path with no more than $3d$ edges between every node in the first part and every node in the third part, and this path can be locally computed. A packet initially at node $\langle \ell, x \rangle$ with destination $\langle \ell', x' \rangle$ can take an arbitrary path forward to level d. By bit comparison, the packet is then led to the node $\langle 2d, x' \rangle$, and then by the direct edges $(\langle k, x' \rangle, \langle k+1, x' \rangle)$, the packet reaches its destination.

Each packet p with destination level q is assigned a fixed rank during the routing defined as $\mathrm{rank}(p) = q - 2d \in [d]$. For each node v in $G(r,s,d,k)$, we define the *median* of v to be the $\frac{r \cdot s}{2}$-largest rank in v if there are at least $\frac{r \cdot s}{2}$ packets in v and -1 otherwise.

2.5. Description of the global protocol

We partition the $rs \cdot N$ messages into 1566 batches such that no more than $rs \cdot m/1566$ messages from each batch that have the same destination level traverse any m-router in the MBF-levels. This can be done by declaring only those packets to be active at level j if their destination is in the set $\{z \mid z = j - 1 \pmod{1566}\}$. For the purpose of analysis we assume that all batch j packets have been routed before transmitting batch $j+1$ packets, and we now concentrate on the routing of one batch.

Nodes at even levels of $G(r,s,d,k)$ are active in odd phases, nodes at odd levels are active in even phases. In one phase, the following routing strategies are performed in the three different parts of $G(r,s,d,k)$:

2.5.1. One Phase within the First d Levels Consider an n-router whose inputs are active. Let ρ denote the minimal rank such that the num-

ber of packets with rank $\geq \rho$ stored in the input and output nodes of that n-router is at most $\frac{1}{522}rsn$. We perform the following two subphases.

The task of the *balancing phase* is to distribute the packets in such a way among the input nodes that there are only very few nodes with more than $\frac{r \cdot s}{4}$ packets with rank $\geq \rho$. Analogous to Proposition 2.3, this can be obtained if each input node sends out its $p \leq r \cdot s$ packets along the edges of the s-ary n-distributor with numbers 1 to $\lceil \frac{p}{r} \rceil$ such that packets with higher rank get edges with lower numbers.

The *delivery phase* consists of four steps. Its task is to approximately sort the packets according to their destination level (by moving packets with rank $\geq \rho$ forward and, maybe, in exchange packets backwards). For each input node, only the $\frac{r \cdot s}{4}$ packets with highest ranks are declared active. In the first step each input node sends a request message to $\frac{s}{2}$ suitably chosen output nodes. Each output node sends its median back to all input nodes that sent it a request. The input node then distributes its packets among the outgoing links such that packets with higher rank are preferred and (up to) r packets are sent along a link if their ranks are larger than the median. If the sum of the packets already stored at an output node and the packets it receives from the input nodes exceeds $r \cdot s$ then the output node sends in exchange to the new packets old packets back preferring packets with lower rank.

2.5.2. One Phase within the Second d Levels Consider an (s, m, k)-router whose inputs are active. Let ρ be defined for the (s, m, k)-router as for the n-sorter above. A phase consists of the following three subphases.

The task of the *balancing phase* is to distribute the packets in such a way among the input nodes that there are only very few nodes with more than $\frac{r \cdot s}{2}$ packets with rank $\geq \rho$. Analogous to Proposition 2.3, this can be obtained if each input node sends out its $p \leq r \cdot s$ packets along the edges of the s-ary n-distributor with numbers 1 to $\lceil \frac{p}{r} \rceil$ such that packets with higher rank get edges with lower numbers.

The task of the *placement phase* is to distribute the packets in such a way among the input nodes that there are only very few nodes having more than $\frac{r \cdot s}{4k}$ packets with rank $\geq \rho$ that have to be sent to the same output set. Analogous to Propositions 2.4 and 2.7, this can be obtained if all packets stored at input nodes are distributed among the s edges of the m-distributor with the help of a suitably chosen assignment graph preferring packets with higher ranks, before sending them out.

The *delivery phase* consists of four steps. Its task it to send as many packets as possible to output sets prescribed by their destinations. For each input node, only the first $\frac{r \cdot s}{4k}$ packets to each of the k output sets are declared

active, preferring packets with higher rank. In the first step each input node sends a request message to $\frac{s}{2k}$ suitably chosen output nodes within each output set to which it has messages to transmit. Each output node sends its median back to all input nodes that sent it a request. The input node then distributes its packets among the outgoing links such that packets with higher rank are preferred and (up to) r packets are sent along a link if their ranks are larger than the median. If the sum of the packets already stored at an output node and the packets it receives from the input nodes exceeds $r \cdot s$ then the output node sends in exchange to the new packets old packets back preferring packets with lower rank.

2.5.3. One Phase within the Last d Levels A phase simply consists of forwarding the packets along the $\frac{s}{2}$ edges for each active node.

2.6. Analysis of the global protocol

We first analyze the routing of one batch. Consider a subgraph connecting two levels in $G(r,s,d,k)$ in a phase in which the inputs of the subgraph are transmitting messages to the outputs. (For the first d levels this would be an $(s,m,1)$-router, for the next d levels any (s,m,k)-router, and for the last d levels any two consecutive levels with active upper level.) Let $q \in [d]$ be fixed. Further let x_1 (resp. y_1) be the total number of packets with rank q or $q+1$ that are stored in the input nodes (resp. output nodes) at the beginning of that phase. Let x_2 (resp. y_2) be the number of packets with rank $\geq q+2$ stored in the input nodes (resp. output nodes) at the beginning of this phase. Moreover, let x' denote the total number of packets with rank q that are stored in the input nodes at the end of the phase. Then we can show the following lemma.

Lemma 2.13.

$$x' \leq \frac{400 \log s}{\sqrt{s}}(x_1 + y_1) + x_2 + y_2 \ .$$

Proof. The result trivially holds for the last d levels. Since the routing strategy in the first d levels is similar the strategy in the second d levels and makes use of $(s,m,1)$-routers, it remains to prove the inequality above for any (s,m,k)-router in the second d levels, $k \in \{1,\ldots,\sqrt{s}\}$.

Let $\epsilon \leq \frac{1}{1566}$. We have to distinguish between two cases. If $x_2+y_2 > \epsilon rsm$ then it immediately follows from the choice of the packets participating in one batch that $x' \leq x_2+y_2$. Suppose in the following that $x_2+y_2 \leq \epsilon rsm$. Then it holds that $x_1+y_1+x_2+y_2 \leq 3\epsilon rsm \leq \frac{1}{522} rsm$. Since the packets with higher

ranks are preferred in our protocol, we can apply Lemma 2.2 with $x = x_1 + x_2$ and $y = y_1 + y_2$ to the protocol above to get that $x' \leq \frac{400 \log s}{\sqrt{s}}(x_1 + y_1 + x_2 + y_2)$. Combining both cases yields the lemma. ∎

Let $X_i^t(k)$ denote the number of packets with rank k in level i after the execution of phase t. Then the following theorem can be shown by using a potential function.

Theorem 2.14. *There is a deterministic multi-port scheme on an extended r-replicated s-ary MBF of size N that routes any global $r \cdot s$-relation in $O(\log_s N)$ steps if s is sufficiently large.*

Proof. We analyze the progress of the routing algorithm in terms of a potential function. Let $w = s^{1/4}/(\log s)^{1/2}$. The *potential* of a packet with rank k after phase t is w^i if after the execution of that phase the packet is in level $2d - i + k$ of the network. (When a packet reaches its destination its potential is 1.) Let $\Phi(t)$ denote the sum of the potentials of the n packets after phase t, that is,

$$\Phi(t) = \sum_{k=0}^{d-1} \sum_{i=0}^{2d+k} X_i^t(k) w^{2d-i+k} \quad .$$

Clearly $\Phi(0) \leq rs \cdot N \cdot w^{3d}$, and routing a batch terminates at the first phase τ such that $\Phi(\tau) < r$.

Let $f(s) = \frac{400 \log s}{\sqrt{s}}$. Assume that t and i are even. Then by Lemma 2.13

$$X_i^{t+1}(k) \leq f(s)(X_i^t(k) + X_{i+1}^t(k) + X_i^t(k+1) + X_{i+1}^t(k+1)) + \sum_{\ell=k+2}^{d-1} (X_i^t(\ell) + X_{i+1}^t(\ell))$$

and

$$X_{i+1}^{t+1}(k) \leq X_i^t(k) + X_{i+1}^t(k) \quad .$$

And after the next phase

$$\begin{aligned} X_i^{t+2}(k) &\leq X_{i-1}^{t+1}(k) + X_i^{t+1}(k) \\ &\leq X_{i-2}^t(k) + X_{i-1}^t(k) + \\ &\quad f(s)(X_i^t(k) + X_{i+1}^t(k) + X_i^t(k+1) + X_{i+1}^t(k+1)) + \\ &\quad \sum_{\ell=k+2}^{d-1} (X_i^t(\ell) + X_{i+1}^t(\ell)), \end{aligned} \tag{4}$$

and

$$
\begin{aligned}
X_{i+1}^{t+2}(k) &\le f(s)\left(X_{i+1}^{t+1}(k)+X_{i+2}^{t+1}(k)+X_{i+1}^{t+1}(k+1)+X_{i+2}^{t+1}(k+1)\right)+\\
&\qquad \sum_{\ell=k+2}^{d-1}\left(X_{i+1}^{t+1}(\ell)+X_{i+2}^{t+1}(\ell)\right)\\
&\le f(s)[\, X_i^t(k)+X_{i+1}^t(k)+\\
&\qquad f(s)\left(X_{i+2}^t(k)+X_{i+3}^t(k)+X_{i+2}^t(k+1)+X_{i+3}^t(k+1)\right)+\\
&\qquad \sum_{\ell=k+2}^{d-1}\left(X_{i+2}^t(\ell)+X_{i+3}^t(\ell)\right)+\\
&\qquad X_i^t(k+1)+X_{i+1}^t(k+1)+f(s)\left(X_{i+2}^t(k+1)+\right.\\
&\qquad \left. X_{i+3}^t(k+1)+X_{i+2}^t(k+2)+X_{i+3}^t(k+2)\right)+\\
&\qquad \sum_{\ell=k+3}^{d-1}\left(X_{i+2}^t(\ell)+X_{i+3}^t(\ell)\right)\,]+\sum_{\ell=k+2}^{d-1}[\, X_i^t(\ell)+X_{i+1}^t(\ell)+\\
&\qquad f(s)(X_{i+2}^t(\ell)+X_{i+3}^t(\ell)+X_{i+2}^t(\ell+1)+X_{i+3}^t(\ell+1))+\\
&\qquad \sum_{\ell'=\ell+2}^{d-1}\left(X_{i+2}^t(\ell')+X_{i+3}^t(\ell')\right)\,]\,.
\end{aligned}
\tag{5}
$$

Plugging these bounds into the potential function and applying the equation

$$X_{i+j}^t(k+\ell)w^{2d-i+k}=w^{j-\ell}\cdot X_{i+j}^t(k+\ell)w^{2d-(i+j)+(k+\ell)}$$

yields

$$
\begin{aligned}
\Phi(t+2) &\le \sum_{k=0}^{d-1}\left[\sum_{\substack{0\le i\le 2d+k,\\ i \text{ even}}}(4)\cdot w^{2d-i+k}+\sum_{\substack{0\le i\le 2d+k,\\ i \text{ odd}}}(5)\cdot w^{2d-i+k}\right]\\
&\le \sum_{k=0}^{d-1}\sum_{\substack{0\le i\le 2d+k,\\ i \text{ even}}}\left[\left(\frac{1}{w^2}+f(s)\left(1+\frac{1}{w}\right)+\sum_{j=2}^{d-1}\left(\frac{1}{w}\right)^j\right)+\right.\\
&\qquad f(s)\left(1+\frac{1}{w}\right)\left(\frac{1}{w}+f(s)(w+1)+\sum_{j=1}^{d-1}\left(\frac{1}{w}\right)^j\right)+\\
&\qquad \left.\sum_{j=0}^{d-1}\left(\frac{1}{w^{3+j}}+f(s)\left(\frac{1}{w^{1+j}}+\frac{1}{w^{2+j}}\right)+\sum_{\ell=3}^{d-1}\left(\frac{1}{w}\right)^{\ell+j}\right)\right]\cdot X_i^t(k)w^{2d-i+k}+
\end{aligned}
$$

$$\sum_{k=0}^{d-1} \sum_{\substack{0\le i\le 2d+k,\\ i \text{ odd}}} \left[\left(\frac{1}{w} + f(s)(w+1) + \sum_{j=1}^{d-1} \left(\frac{1}{w}\right)^j \right) + f(s)\left(1+\frac{1}{w}\right)\left(1+f(s)(w^2+w)+\sum_{j=0}^{d-1}\left(\frac{1}{w}\right)^j\right) + \sum_{j=0}^{d-1}\left(\frac{1}{w^{2+j}} + f(s)\left(\frac{1}{w^j}+\frac{1}{w^{1+j}}\right) + \sum_{\ell=2}^{d-1}\left(\frac{1}{w}\right)^{\ell+j}\right)\right] \cdot X_i^t(k) w^{2d-i+k}$$

$$\le \sum_{k=0}^{d-1} \sum_{i=0}^{2d+k} \left[\left(\frac{1}{w} + f(s)(w+1) + \sum_{j=1}^{d-1} \left(\frac{1}{w}\right)^j \right) + f(s)\left(1+\frac{1}{w}\right)\left(1+f(s)(w^2+w)+\sum_{j=0}^{d-1}\left(\frac{1}{w}\right)^j\right) + \sum_{j=0}^{d-1}\left(\frac{1}{w^{2+j}} + f(s)\left(\frac{1}{w^j}+\frac{1}{w^{1+j}}\right) + \sum_{\ell=2}^{d-1}\left(\frac{1}{w}\right)^{\ell+j}\right)\right] \cdot X_i^t(k) w^{2d-i+k} \quad .$$

Thus the potential function is decreased by at least

$$\begin{aligned} &\left(\frac{1}{w} + f(s)(w+1) + \sum_{j=1}^{d-1} \left(\frac{1}{w}\right)^j \right) + \\ &f(s)\left(1+\frac{1}{w}\right)\left(1+f(s)(w^2+w)+\sum_{j=0}^{d-1}\left(\frac{1}{w}\right)^j\right) + \\ &\sum_{j=0}^{d-1}\left(\frac{1}{w^{2+j}} + f(s)\left(\frac{1}{w^j}+\frac{1}{w^{1+j}}\right) + \sum_{\ell=2}^{d-1}\left(\frac{1}{w}\right)^{\ell+j}\right) \\ &= O(w^{-1}) \end{aligned}$$

every two phases, and for $\tau = O(\log_s N)$, $\Phi(\tau) < r$. Since there are only $O(1)$ batches to route the theorem follows. ∎

Combining the theorem with the results in [36] if s is not sufficiently large yields Theorem 1.2.

3. The "Routing via simulation" technique

In this section we present a very efficient deterministic routing protocol for arbitrary networks H. The idea of the protocol is that, for any permutation

routing problem in H, we route the packets to their destinations by simulating routing in a suitably chosen extended s-ary multibutterfly embedded in H. For this we use the simulation strategy described in Section 3.1.

3.1. Network emulations using 1-many embeddings

Consider the problem of simulating one routing step of an arbitrary network G by a network H for the case that the size of G is smaller than the size of H. We start with describing how to embed G in H. In order to simplify the construction, let H be a network of size n, and G be a network of size m with at most $n/2$ edges. Let $d_1,\ldots,d_m$ be the degree sequence of G, i.e., d_i is the degree of node i in G. Then $\sum_{i=1}^{m} d_i \leq n$. Our strategy is to partition the nodes of H into clusters $C_1,\ldots,C_m$ such that for all $i \in \{1,\ldots,m\}$ cluster C_i consists of d_i nodes representing d_i copies of node i in G. For this we choose an arbitrary spanning tree T in H. Let r be an arbitrary node in T. We mark the nodes in T with numbers in $\{1,\ldots,m\}$ starting with r by calling Mark$(1,true,r)$:

Algorithm Mark(i,f,v):

i: number of nodes already marked
f: boolean variable indicating whether father has been marked
v: actual node to be considered

if $f = false$ **then**
  mark v with the number ℓ obeying $\sum_{j=1}^{\ell-1} d_j < i \leq \sum_{j=1}^{\ell} d_j$
  set $i = i+1$
  for every son w of v: call Mark$(i,true,w)$
else
  for every son w of v: call Mark$(i,false,w)$
  mark v with the number ℓ obeying $\sum_{j=1}^{\ell-1} d_j < i \leq \sum_{j=1}^{\ell} d_j$
  set $i = i+1$
return the value of i

Basically, the algorithm ensures that on a pass downwards through the tree only every second node is marked such that afterwards on a pass upwards the other half of the nodes can be marked. Consider two nodes marked as ith and $(i+1)$st node. Then on a pass downwards, the cases shown in Figure 3. can occur.
On a pass upwards, the cases shown in Figure 4. can occur.

As can be seen, the worst case that can happen with this strategy is that two consecutively marked nodes have a distance of 3 (by changing from one

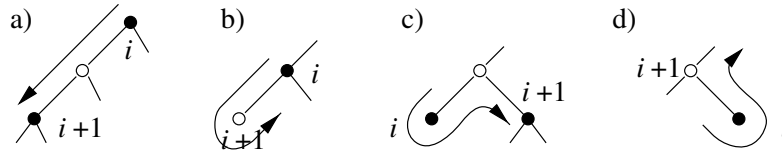

Fig. 3. Alternatives for a pass downwards.

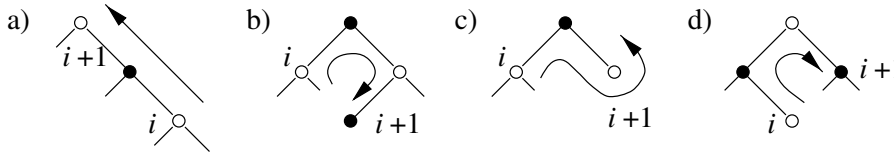

Fig. 4. Alternatives for a pass upwards.

subtree into another). Hence nodes belonging to a cluster i have at most a distance of $3d_i$ from each other. Let the nodes of each cluster be connected by an Euler tour along edges in T. (An Euler tour in a tree is defined as a directed cycle that uses any edge in T at most once in every direction.) Then a link is only used by two Euler tours: the Euler tour belonging to that cluster that was built while traversing that link downwards, and the Euler tour belonging to that cluster that was built while traversing that link upwards. Hence the following two results hold.

(a) the Euler tour of cluster i has length at most $6d_i$ for all $i \in \{1, \ldots, m\}$, and
(b) the maximal number of Euler tours that share the same link is two.

We further want to simulate every edge in G by a path in H that connects the nodes simulating its endpoints. Let R be the routing number of H. Since our clustering allows the endpoints of edges in G to be distributed in H such that every node in H has to simulate at most one endpoint, there is a path collection in H for simulating the edges in G with congestion at most R and dilation at most R (see Remark 1.1).

Consider now the problem of simulating an arbitrary routing step in G. Clearly, any routing step can be extended to the situation that along every edge in G a packet has to be sent. This event can be simulated in the following way by H.

- Moving the packets to the nodes simulating the endpoints of the edges they want to use in G: This can be done by sending the packets along an Euler tour connecting the nodes of the respective cluster in T. Because of

(b) this can be coordinated among the clusters deterministically in time $O(\max_i d_i)$ using only constant size buffers.

- Moving the packets along an edge in G: This can be done by sending the packets along the paths simulating edges in G. Since these paths have congestion at most R and dilation at most R, this can be done deterministically in time $O(R)$ using only constant size buffers by applying the offline protocol by Leighton *et al.* [24].

If we restrict the maximum degree in G to be $O(R)$, we get the following result.

Lemma 3.1. *Any network H of size n with routing number R can simulate any routing step in a network G with degree $O(R)$ and $O(n)$ edges in $O(R)$ steps, using only constant size buffers.*

3.2. A deterministic (non-compact) routing protocol

Consider an arbitrary network H of size n with routing number R. In a preprocessing phase, we embed an extended R-ary multibutterfly of size approximately n/R in H such that every node in the multibutterfly is simulated by approximately R nodes in H. According to Lemma 3.2 such a multibutterfly exists.

Lemma 3.2. *For any $\sqrt{s}, n \geq 2$ there exist $k \in \{1, \ldots, \sqrt{s}\}$ and $d \geq 1$ such that the number m of nodes in the (s,d,k)-XBF is bounded by $n/2 \leq m \leq n$.*

Proof. According to the definition of the extended multibutterfly, the number of nodes in the (s,d,k)-XBF is $dk\cdot\sqrt{s}^{d-1}$. Choose $d \geq 1$ and $k \in \{1, \ldots, \sqrt{s}\}$ in such a way that the number m of nodes gets maximal under the restriction that $m \leq n$. We distinguish between two cases.

- $k < \sqrt{s}$: Then $d\cdot(k+1)\cdot\sqrt{s}^{d-1} > n$ and therefore $d\cdot k\cdot\sqrt{s}^{d-1} \geq \frac{1}{2}d\cdot(k+1)\cdot\sqrt{s}^{d-1} > n/2$.
- $k = \sqrt{s}$: Then $(d+1)\cdot 1\cdot\sqrt{s}^{d} > n$ and therefore $dk\cdot\sqrt{s}^{d-1} \geq \frac{1}{2}(d+1)\cdot\sqrt{s}^{d} > n/2$.

This yields the lemma. ∎

In the following, we assume that there exists an R-ary multibutterfly that has exactly n/R nodes. In this case we want to assign exactly R nodes of H to any node of the multibutterfly. (If this is not possible, then Lemma 3.2 implies that an R-ary multibutterfly can be chosen such that we need clusters of size at most $2R$ to assign all nodes in H to nodes of the multibutterfly.)

In order to partition the nodes of H into clusters of size R, we choose an arbitrary spanning tree T in H and apply the clustering strategy on T described in Section 3.1. Let the nodes of each cluster be connected by an Euler tour along edges in T. This ensures that

(a) every Euler tour has a length of at most $6R$, and
(b) the maximal number of Euler tours that share the same link is two.

Further we distribute the paths simulating edges of the multibutterfly among the nodes in H in such a way that every node is the endpoint of at most some constant number of paths and simulates at most one in- and outgoing edge for each distributor and splitter. If we now want to route any permutation in H, we can transform this into the problem of routing any R-relation in the R-ary multibutterfly. Hence, in order to route a permutation in H, we can choose to perform a deterministic step-by-step simulation of routing an R-relation in the multibutterfly. With this strategy we can prove the following result.

Theorem 3.3. *Let H be an arbitrary network of size n with routing number R. Then there is a deterministic online protocol that routes any permutation in time $O(\log_R n \cdot R)$, using only constant size buffers.*

Proof. Each step of the multibutterfly can be simulated in the following way in H.

- **Assigning XBF-edges to the packets:**
  In case that we have to simulate a balancing phase, this can be done by first sorting the packets in the nodes of each cluster according to their rank (note that the rank of a packet depends on its destination level in the XBF, see Section 2.5). Since only a constant number of Euler tours share the same link, this can be done in time $O(R)$ with constant size buffers, using Odd-Even Transposition Sort [16]. Afterwards, for all $i \in \{1, \ldots, R\}$ the packet with ith largest rank is sent along the Euler tour of its cluster until it reaches the node that simulates edge i of the R distributor edges leaving that cluster. Clearly, this can also be coordinated among the clusters in time $O(R)$.
  In case that we have to simulate a placement phase, we again first sort the packets in the nodes of each cluster according to their rank. As noted above, this takes $O(R)$ steps. Next the $R/2$ packets with highest ranks have to be assigned to suitably chosen distributor edges. For this, the nodes first count how many of these packets want to be sent to any of the at most $\sqrt{R}$ output sets. Using this information, the nodes compute which packet to forward along which distributor edge. (Note that this

can be done by each node with the help of an algorithm presented by Vazirani [38] that runs in $O(R^{3/2})$ time to find an assignment of packets to edges. Since a calculation step can be usually performed much faster than a communication step, we will not consider the time for computing such an assignment.) This information is used to distribute the packets among the nodes of the cluster. Clearly, the counting and distribution of packets can be performed in $O(R)$ steps for every cluster, using only constant size buffers.

In case that we have to simulate the first step of a delivery phase, we generate a request packet for each node simulating the endpoint of a splitter edge. After all answers of the requests are received (for this we need the routing strategy below), the packets in each cluster are first sorted according to the output set they want to reach and their rank, and then delivered among the nodes that received a positive answer, preferring packets with higher rank. As above, the sorting and distribution of the packets can be performed in $O(R)$ steps for every cluster, using only constant size buffers.

- **Moving each packet along its assigned XBF-edge:**
  According to the definition of the routing number, the edges of the XBF can be simulated by a path collection in H with congestion $O(R)$ and dilation at most R. Since at most one packet is sent along each of these paths, there is an offline protocol according to [24] that routes packets along these paths in time $O(R)$ using only constant size edge buffers.

Combining these results with Theorem 1.2 yields the theorem. ∎

It remains to show how to extend this protocol to a deterministic compact routing protocol.

4. Deterministic compact routing

In this section we present a compact deterministic routing protocol for arbitrary networks H. For this let us recall the general approach of the routing via simulation technique.

Let H be an arbitrary network of size n with routing number R. Suppose, we want to simulate a network G of size $m = O(n/R)$ and degree $s = O(R)$ by H. In order to simplify the description, let us assume in the following that $m = n/R$. Let H be partitioned into clusters of size R as described in Section 3.1, each representing a node in G. Furthermore let $\mathcal{E}$ represent the set of all Euler tours connecting the nodes of the clusters and $\mathcal{P}_G$ be a path collection in H which contains for each edge $\{u, v\}$ in G a path from

the cluster representing u to the cluster representing v. In the following, we want to show how to design compact routing structures for $\mathcal{E}$, $\mathcal{P}_G$, and the offline protocol for simulating one routing step in G.

4.1. Selecting suitable routing paths

In order to route packets between two nodes within a cluster, we simply send them along the corresponding Euler tour as it was done in the deterministic non-compact protocol. It remains therefore to show the existence of routing paths for $\mathcal{P}_G$ with low congestion and dilation. For this we need the following lemma.

Lemma 4.1. *There exists a simple path collection in H for simulating all edges in G with dilation at most R and congestion $O(s+\log R)$.*

Proof. We distinguish between three cases. If $s \geq R$ then choose each node in H to be the endpoint of at most $\lceil s/R \rceil$ paths. Thus the problem of establishing a path for each edge in G reduces to the problem of finding an efficient path collection for an arbitrary $\lceil s/R \rceil$-relation routing problem. Because of the definition of R, there exists a simple path collection for any such problem with congestion at most $R \cdot \lceil s/R \rceil$ and dilation at most R.

Consider now the case that $\log R \leq s < R$. Then we want to show with the help of the Lovász Local Lemma (see [3], p.55) that there exists a simple path collection $\mathcal{P}_G$ in H with congestion $O(s)$ and dilation at most R.

Lemma 4.2. *(Lovász) Let $A_1, \ldots, A_m$ be a set of "bad" events, each A_i occuring with probability at most p and depending on at most b other events in $\{A_1, \ldots, A_m\}$. If $ep(b+1) < 1$, then with probability greater than zero no bad event occurs.*

For any connection that has to be established between any cluster C_1 and C_2, $\lfloor R/s \rfloor$ pairs $\{u, v\}$ of nodes are chosen as *candidates*, $u \in C_1$, $v \in C_2$. These candidates can be chosen such that each node belongs to at most one candidate. Hence there exists a collection of simple paths, one for each candidate, with congestion at most R and dilation at most R. Now consider the random experiment of choosing randomly for each connection one of the $\lfloor R/s \rfloor$ paths representing its candidates and eliminating the rest. We associate a bad event to each edge e in H. The bad event for edge e is that more than k surviving paths contain e (k will be determined later). In order to show that there is a way of choosing the candidates such that no bad event occurs, we need to bound the dependence b among the bad events and the probability p of each individual bad event occurring.

The dependence calculation is straightforward. Whether or not a bad event occurs depends solely on the selection of the candidates that pass through the corresponding edge. Since at most R candidates pass through an edge, and each of these candidates belongs to a set of $\lfloor R/s \rfloor$ candidates for a connection, each having a length of at most R, the dependence b of the bad events is at most $R \cdot \lfloor R/s \rfloor \cdot R$.

Next we compute the probability of each bad event. Let p be the probability of the bad event corresponding to edge e. Then

$$p \leq \binom{R}{k} \left(\frac{1}{\lfloor R/s \rfloor} \right)^k \leq \left(\frac{\mathrm{e} \cdot R}{k \cdot \lfloor R/s \rfloor} \right)^k .$$

Since $s \geq \log R$, for $k \geq 4\mathrm{e} \cdot s$ the product $\mathrm{e}p(b+1)$ is less than 1, and thus, by the Lovász Local Lemma, there is a choice of candidates such that the congestion is $O(s)$.

If $s < \log R$ then we choose $\lceil \log R/s \rceil$ random paths from the candidates for any connection to survive and eliminate the rest. Similar to above, we can show with the help of the Lovász Local Lemma that there is a choice of candidates such that the congestion is $O(\log R)$. ∎

4.2. Design of compact routing tables

In this section we present a method to design compact routing tables for storing $\mathcal{E}$ and $\mathcal{P}_G$ in the nodes in H. First we consider the problem of storing $\mathcal{P}_G$.

Lemma 4.3. *Let d be the degree of H. Then H needs at most $O((s + \log R)d \log d)$ space in each vertex for storing $\mathcal{P}_G$.*

Proof. Let C be the congestion and D be the dilation of $\mathcal{P}_G$. Clearly, every path in $\mathcal{P}_G$ shares its links with at most $C \cdot D$ other paths in $\mathcal{P}_G$. Suppose $H' = (V', E')$ is a graph in which each node represents a path in $\mathcal{P}_G$ and nodes $x, y \in V'$ are connected with each other if their respective paths share a link in H. Then H' has a degree of at most $d' = C \cdot D$. Clearly, $d' + 1$ colors suffice to color every d'-regular graph in such a way that no two adjacent vertices have the same color. Therefore it is possible to attach numbers to the paths in $\mathcal{P}_G$ out of $[C \cdot D + 1]$ in such a way that no two paths in $\mathcal{P}_G$ with a common link have the same number.

Let $\psi : \mathcal{P}_G \to [C \cdot D + 1]$ be the function that assigns a number to all paths in $\mathcal{P}_G$ such that the condition above is fulfilled. Then we choose the following strategy to store routing information.

Consider any node v in H of degree d_v. Let the edges of v be numbered from 0 to d_v-1 and $\mathcal{P}_v$ be the set of all paths in $\mathcal{P}_G$ that have their endpoint at v. In order to store $\mathcal{P}_G$ we need the following two tables.

- $T_{v,1}:\mathcal{P}_v \to [d_v]\times[C\cdot D+1], p\to(e,\psi(p))$ maps each path in $\mathcal{P}_v$ to a suitable color and the first edge e used by this path in H.
- $T_{v,2}:[d_v]\times[C\cdot D+1]\to[d_v]\cup\{\emptyset\}$ is arranged such that $T_{v,2}(e,k)$ contains the edge the path with number k entering v via e uses to leave v.

Since $s=O(R)$, the paths in $\mathcal{P}_G$ can be distributed among the nodes in such a way that $|\mathcal{P}_v|$ is a constant. Therefore $T_{v,1}$ can be easily implemented with constant lookup time using only $O(\log(d\cdot C\cdot D))$ bits. Furthermore, it takes at most $O(d\cdot C\cdot D\cdot\log d)$ bits to store $T_{v,2}$. This can be improved by using a theorem shown by Schmidt and Siegel in [33].

Theorem 4.4. *For any set of n elements belonging to the universe $[1,m]$, an $O(1)$-time perfect hash function can be specified by $O(n+\log\log m)$ bits.*

With the help of a perfect hash function, we can therefore reduce the size of $T_{v,2}$ to $O(d\cdot C\log d)$ bits in such a way that we can still evaluate $T_{v,2}$ in constant time by a function that needs $O(d\cdot C+\log\log(d\cdot C\cdot D))$ bits. Altogether this results in routing tables of size

$$O(C\cdot d\log d+\log(d\cdot C\cdot D))=O((s+\log R)d\log d)$$

per node for storing $\mathcal{P}_G$, because according to Lemma 4.1 we have $D\le R$ and $C=O(s+\log R)$. ∎

Consider now the problem of storing the clusters in the nodes of H. For this we can show the following lemma.

Lemma 4.5. *Let d be the degree of H. Then each node needs at most $O(d\log d+\log R)$ space to store all Euler tours traversing it.*

Proof. According to Section 3.1, every Euler tour shares its edges with at most $C=O(R)$ other Euler tours. Then $C+1$ colors suffice to color the Euler tours in such a way that no two Euler tours with the same color share an edge. Thus we can choose the following strategy to store routing information.

Consider any node v in H of degree d_v. First we have to store in v the color of the Euler tour v belongs to, and the edge the Euler tour uses to leave v. This takes $\log R+\log d+O(1)$ bits. Furthermore we need the following table.

- $T_{v,3}:[d_v]\times[C+1]\to[d_v]$ is arranged such that $T_{v,3}(e,k)$ contains the next edge the Euler tour with number k entering v via e uses to leave v.

Clearly, it takes at most $O(d \cdot C \cdot \log d)$ bits to store $T_{v,3}$. Since the Euler tours have constant congestion, we can apply perfect hashing techniques (see Theorem 4.4) to reduce the size of $T_{v,3}$ to $O(d \log d)$ bits in such a way that we can still evaluate $T_{v,3}$ in constant time by a hash function that needs $O(d + \log\log(d \cdot C))$ bits. Summing over all space requirements yields the lemma. ∎

The tables described in Lemma 4.3 and Lemma 4.5 can be used in the following way. Suppose, we want to send a packet p along a path P in $\mathcal{E}$ or $\mathcal{P}_G$. Let p be currently stored at the endpoint v of P in H. We have to distinguish between two cases.

If P is a path connecting two different clusters then we use the tables described in Lemma 4.3. First, p gets the color c and the first edge e of the path P by accessing $T_{v,1}$. The packet chooses to traverse e and stores the color c in its routing information. Let e' be the last edge p used so far to reach some node u. With the help of $T_{u,2}$, e' and its actual color, the packet determines the edge it has to traverse next in H. p continues to access $T_{u,2}$ for each node u it visits until it reaches the other endpoint of P (in this case, we have $T_{u,2}(e', c) = \emptyset$).

If P is a path connecting two nodes within one cluster, we use the table described in Lemma 4.5. First, v provides p with the color c of the cluster it belongs to and the next edge e of the Euler tour p has follow in that cluster. The packet chooses to traverse e and stores the color c in its routing information. Let e' be the last edge p used so far, and p be currently stored in node u. With the help of $T_{u,3}$, e' and its actual color, the packet determines the edge it has to traverse next along the Euler tour. It continues to access $T_{u,3}$ for each node u it visits until it reaches the other endpoint of P.

4.3. Compact offline routing

In order to bound the space requirements for the offline protocol we want to use to simulate a routing step in G by H, we need the following lemma.

Lemma 4.6. *Consider an arbitrary simple path collection with congestion C and dilation D such that the sources of the paths are disjoint and all nodes have degree at most d. Suppose that along each path p packets have to be sent. Then there exists an offline protocol that can route all packets in time $O(p \cdot C + D)$ if constant size buffers are available at each edge and $\Theta(d \cdot C + \log p)$ space is available at each node for storing the protocol.*

Proof. Let the packets be divided into p batches such that each batch contains exactly one packet for every path.

Consider first the case that $C \geq D$. Then we can use the offline protocol described in [24] to route any batch in time $O(C+D) = O(C)$, using only constant size edge buffers. Hence altogether we need time $O(p \cdot C)$ to route all packets. Since every batch uses the same collection of paths, we can use the same offline protocol for every batch. We therefore only need to store in the nodes the offline protocol for one batch and a counter for the number of batches that have already been routed. Clearly, the counter needs $O(\log p)$ space. Every packet waits at most $O(C)$ time steps in its source before it traverses its first edge. This needs $O(\log C)$ space, because we assume that the sources of the paths are disjoint. Since each packet only has to wait a constant number of steps in any buffer once it has started (see [24]), each edge needs at most $O(C)$ space to coordinate arriving packets by using a table with entries for every time point of the protocol. An entry is 0 if no packet arrives at this time and otherwise contains the number of steps a packet arriving at this time point has to wait. As every node has degree at most d, $\Theta(d \cdot C + \log p)$ space suffices in each node to execute the offline protocol for all batches.

Consider now the case that $C < D$. Let all paths be divided into subpaths of length at least C and at most $2C$. (If a path has length below C then it is considered as one single subpath.) Let a subpath be called *intermediate* if it is neither the first nor the last subpath of the path it belongs to. We want to choose an edge in each intermediate subpath such that no edge is chosen more than once. Let us call edges with this property *secure*. In order to show that a secure edge can be assigned to every intermediate subpath for any choice of the subpaths obeying the length constaints above in any path collection, consider the following construction.

Let $G = (V_1, V_2, E)$ be a bipartite graph with V_1 representing the intermediate subpaths and V_2 representing all edges used by the path collection. A node $u \in V_1$ is connected to node $v \in V_2$ if the subpath representing u contains the edge representing v. Since each intermediate subpath has length at least C, all nodes in V_1 have degree at least C. Furthermore, every node in V_2 has degree at most C, because every edge is used by at most C paths and therefore by at most C subpaths. Hence it holds for every subset $U \subseteq V_1$ that $|\Gamma(U)| \geq |U|$. Otherwise there must exist a node in $\Gamma(U)$ with degree at least $C+1$. From Theorem 2.6 it follows that there must exist a matching of size $|V_1|$ in G. Thus for each intermediate path a secure edge can be chosen.

For every path, let its first edge be the secure edge of its first subpath. Consider now the situation that each secure edge has one packet in its buffer,

and every packet has to be sent to the next secure edge (or the destination) on its respective path. This routing problem has congestion $O(C)$ and dilation $O(C)$. Hence the offline protocol described in [24] can be used to route all packets in time $O(C)$, using only constant size edge buffers at any time of the execution.

Our goal now is to interpret the secure edges as intermediate destinations, and to send the batches of packets along these intermediate destinations in a pipelined fashion using the offline protocol above, starting with batch 1-packets followed by batch 2-packets, and so on. If we use this strategy to route the p batches of packets along their respective paths, the runtime and requirements for the buffer size of the offline protocol above imply that the overall runtime is bounded by $O(p \cdot C + D)$, using only constant size edge buffers. Analogous to the case $C \geq D$ each node needs $O(d \cdot C + \log p)$ space to execute the offline protocol for all batches. ∎

4.4. A deterministic compact routing protocol

In this section we finally prove the Main Theorem. Let H be an arbitrary network with n nodes and routing number R. Given any $\sqrt{s}, n \geq 2$, let $G(r,s,n)$ denote the (r,s,d,k)-XBF whose size n' is closest to n. According to Lemma 3.2 it holds that $n/2 \leq n' \leq n$.

Let $s \leq R$. We partition the nodes of H into $\lfloor n/R \rfloor$ clusters of size R using the strategy described in Section 4.1, each simulating a single node in $G(\lfloor \frac{R}{s} \rfloor, s, \frac{n}{R})$. If the size of $G(\lfloor \frac{R}{s} \rfloor, s, \frac{n}{R})$ multiplied by R is less than n then some nodes in H are not assigned to clusters. In this case we only have to increase the size of the clusters by a factor of at most two.

Let us first prove the time bound of the Main Theorem. Since each edge in $G(\lfloor \frac{R}{s} \rfloor, s, \frac{n}{R})$ has $\lfloor \frac{R}{s} \rfloor$ channels, and each node in $G(\lfloor \frac{R}{s} \rfloor, s, \frac{n}{R})$ is simulated by $\Theta(R)$ nodes in H, the channels can be distributed among the nodes such that every node is assigned to at most a constant number of channels. If $s \geq \log R$ then each edge of $G(\lfloor \frac{R}{s} \rfloor, s, \frac{n}{R})$ has a unique path in H. Otherwise, we partition the channels into $\lceil \log R/s \rceil$ sets of equal size and assign each set to one of the $\lceil \log R/s \rceil$ paths simulating an edge in $G(\lfloor \frac{R}{s} \rfloor, s, \frac{n}{R})$. Then each step of the multibutterfly can be simulated in the following way in H.

- **Assigning XBF-channels to the packets:**
  For this we basically use the same strategies as described in Theorem 3.3, with the difference that here we require the packets to be distributed among nodes simulating endpoints of channels instead of endpoints of edges.

- **Moving each packet along its assigned XBF-channel:**
  In order to route the packets along their assigned XBF-channel, consider the packets to be seperated into batches, each batch representing a different channel number (or a set of different channel numbers such that there is one for each path in case that $s < \log R$). Since the Euler tours have constant congestion, it is easy to send the packets batch after batch to the starting point of the path they have to take if for each XBF-edge the nodes simulating its channels are ordered from channel 1 to channel $\lfloor \frac{R}{s} \rfloor$ along their Euler tour, and for $s \geq \log R$ the node simulating channel 1 represents the starting point of the path simulating that XBF-edge. (For $s < \log R$ we evenly distribute the starting points of the paths simulating that XBF-edge among the channel numbers.)
  As shown above, the edges of the XBF can be simulated by a path collection in H that has dilation at most R and congestion $C = O(s + \log R)$. Since at most $O(\frac{R}{C})$ packets are sent along each of these paths, the congestion of the path collection is bounded by $O(R)$. Hence we can use the strategy described in Lemma 4.6 to route the packets in batches along the paths in time $O(R)$, using only constant size buffers.

Combining these results with Theorem 1.2 yields the time bound of the Main Theorem.

It remains to prove an upper bound for the space necessary to store routing information in the nodes and packets. Let d be the degree of H.

- **Storing the embedding of $G(\lfloor \frac{R}{s} \rfloor, s, \frac{n}{R})$ into H:**
  It is easy to see that we can choose cluster sizes and numbers for the nodes in H such that we need $O(\log n)$ bits to store a function h in each node telling it that a node with number x in H simulates node $h(x)$ in $G(\lfloor \frac{R}{s} \rfloor, s, \frac{n}{R})$.
- **Storing the Euler tours:**
  According to Lemma 4.5, $O(d \log d + \log R)$ bits suffice to store a lookup table in such a way that for each cluster the packets can be routed along an Euler tour.
- **Storing the paths simulating edges in the XBF:**
  According to Lemma 4.3, $O((s + \log R) d \log d)$ bits suffice to store a lookup table for the at most $O((s + \log R)d)$ paths crossing each node.
- **Storing routing information in the packets:**
  According to Section 4.2, packets have to be able to store the color of the path they are currently following. Since $\mathcal{P}_G$ has dilation at most R and congestion $O(s + \log R)$, this takes $O(\log(s \cdot R))$ bits. Furthermore, they have to store the color of their current cluster, which takes $O(\log R)$ bits. Note that the rank of a packet can be computed with the help of

its destination address and h and therefore does not need any additional bits in the packets.

- **Storing the XBF structure:**
  Consider a node v in H that is the endpoint of a path simulating an edge in the XBF. If this edge belongs to a distributor, v needs its number, which takes $O(\log s)$ bits. If this edge belongs to a splitter, v needs to know to which output set it leads. This also takes $O(\log s)$ bits. If $s < \log R$, several paths may exist for each XBF-edge. In this case, we need $O(\log\log R)$ bits for each path to specify which set of channels it represents. Hence each node requires $O(\log(s + \log R))$ bits to store the XBF structure.
- **Storing the offline protocol:**
  Because the congestion of $\mathcal{P}_G$ is $O(s + \log R)$, we need at most $O(d(s + \log R))$ bits in each node to store the offline protocol.
- **Storing information about the assignment of paths:**
  Since the placement phase is the only phase that requires working with a special assignment graph, it suffices to consider the space requirements for simulating the placement phase.
  According to Proposition 2.4, $O(s\sqrt{s}\log s)$ space is necessary to store the assignment graph of an s-ary XBF. This can be distributed among the nodes of a cluster such that each node needs at most $O(\sqrt{s}\log s)$ space for storing a part of that assignment graph. Further each node needs $O(\sqrt{s}\log s)$ space to store the number of packets in its cluster that have to be sent to any of the $\sqrt{s}$ output sets. Given a fixed number of packets to each of these output sets, each node that stores nodes of the assignment graph representing one block of size $\lfloor \frac{R}{s} \rfloor$ of these packets sends out packets containing information about edges adjacent to these nodes. Since nodes in the assignment graph representing blocks of packets have constant degree, the number of bits necessary to store information about edges adjacent to any such node of the assignment graph is at most $O(\log s)$. Hence at most $O(s)$ packets have to be sent along the Euler tour to inform all nodes of the respective cluster about the subgraph of the assignment graph for which a maximum matching has to be found. Storing this subgraph requires $O(s \log s)$ bits in each of the nodes. Using Vazirani's algorithm [38], it further takes $O(s \log s)$ space to compute a maximum matching.

Combining all space requirements yields the space bounds of the Main Theorem.

5. Conclusions

The aim of this paper is to show that it is possible to design asymptotically very efficient protocols for compact deterministic routing. We hope that this result will ignite research in the areas of compact and deterministic routing to find not only asymptotically, but also practically efficient protocols. Another open problem is whether the results presented here can still be improved asymptotically. It would furthermore be interesting to prove stronger lower bounds for both areas, since there still is a significant gap between the best known upper and lower bounds for the runtime of deterministic and compact routing protocols.

Acknowledgements. We would like to thank Berthold Vöcking for helpful comments on an early draft of the paper.

References

[1] M. Ajtai, J. Komlós, E. Szemerédi: Sorting in $c \log n$ parallel steps, *Combinatorica* **3** (1983), 1–19.

[2] N. Alon, F. R. K. Chung, R. L. Graham: Routing permutations on graphs via matchings, *SIAM J. on Discrete Mathematics* **7**(3) (1994), 513–530.

[3] N. Alon, P. Erdős, J. Spencer: *The Probabilistic Method*, Wiley Interscience Series in Discrete Mathematics and Optimization, John Wiley & Sons, 1992.

[4] B. Awerbuch, A. Bar-Noy, N. Linial, D. Peleg: Improved routing strategies with succinct tables, *Journal of Algorithms* **11** (1990), 307–341.

[5] B. Awerbuch, D. Peleg: Routing with polynomial communication-space tradeoff, *SIAM J. on Discrete Mathematics* **5** (1992), 151–162.

[6] A. Borodin, J. E. Hopcroft: Routing, Merging, and Sorting on Parallel Models of Computation, *Journal of Computer and System Sciences* **30** (1985), 130–145.

[7] A. Borodin, P. Raghavan, B. Schieber, E. Upfal: How much can hardware help routing? in: *Proc. of the 25th Ann. ACM Symp. on Theory of Computing*, 573–582, 1993.

[8] R. Cypher, C. G. Plaxton: Deterministic sorting in nearly logarithmic time on the hypercube and related computers, *Journal of Computer and System Sciences* **47** (1993), 501–548.

[9] R. Cypher, F. Meyer auf der Heide, C. Scheideler, B. Vöcking: Universal algorithms for store-and-forward and wormhole routing, in: *28th Ann. ACM Symp. on Theory of Computing*, 356–365, 1996.

[10] P. Fraigniaud, C. Gavoille: Optimal interval routing, in: *CONPAR '94 – VAPP VI*, Springer Verlag, LNCS 854, 785–796, 1994.

[11] P. Fraigniaud, C. Gavoille: Local memory requirement for universal routing schemes, in: *8th Ann. ACM Symp. on Parallel Algorithms and Architectures*, 183–188, 1996.

[12] G. N. Frederickson and R. Janardan: Efficient message routing in planar networks, *SIAM Journal on Computing* **19** (1989), 843–857.

[13] G. N. Frederickson and R. Janardan: Space-efficient message routing in c-decomposable networks, *SIAM Journal on Computing* **19**(1) (1990), 164–181.
[14] C. Gavoille: On Dilation of Interval Routing, in: *Symp. on Mathematical Foundations of Computer Science*, 1997.
[15] C. Kaklamanis, D. Krizanc, T. Tsantilas: Tigth Bounds for Oblivious Routing in the Hypercube, *Mathematical Systems Theory* **24** (1991), 223–232.
[16] D. E. Knuth: *The Art of Computer Programming, Volume 3: Sorting and Searching*, Addison-Wesley, 1973.
[17] R. Kráľovič, P. Ružička, D. Štefankovič: *The complexity of shortest path and dilation bounded interval routing,* Technical Report, Comenius University, Department of Computer Science, Bratislava, Slovak Republic, Aug. 1996.
[18] D. Krizanc: Oblivious Routing with Limited Buffer Capacity, *Journal of Computer and System Sciences* **43** (1991), 317–327.
[19] M. Kunde: Optimal Sorting on Multi-Dimensionally Mesh-Connected Computers, in: *Proc. of the 4th Symp. on Theoretical Aspects of Computer Science*, 408–419, 1987.
[20] F. T. Leighton: Tigth bounds on the complexity of parallel sorting, *IEEE Transactions on Computers*, **C-34** (1985), 344–354.
[21] F. T. Leighton: *Introduction to Parallel Algorithms and Architectures: Arrays · Trees · Hypercubes*, Morgan Kaufmann Publishers, San Mateo, CA, 1992.
[22] F. T. Leighton, B. Maggs: Expanders might be practical: fast algorithms for routing around faults in multibutterflies, in: *Proc. of the 30th Ann. IEEE Symp. on Foundations of Computer Science*, 384–389, 1989.
[23] T. Leighton, B. Maggs, A. Ranade, S. Rao: Randomized routing and sorting on fixed-connection networks, *Journal of Algorithms* **17** (1994), 157–205.
[24] T. Leighton, B. Maggs, S. Rao: Packet routing and job-shop scheduling in O(Congestion + Dilation) steps, *Combinatorica* **14** (1994), 167–186.
[25] T. Leighton, S. Rao: An approximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approximation algorithms, in: *Proc. of the 29th Ann. IEEE Symp. on Foundations of Computer Science*, 422–431, 1988.
[26] F. Meyer auf der Heide, C. Scheideler: Space-efficient routing in vertex-symmetric networks, in: *7th Ann. ACM Symp. on Parallel Algorithms and Architectures*, 137–146, 1995.
[27] F. Meyer auf der Heide, C. Scheideler: Routing with bounded buffers and hot-potato routing in vertex-symmetric networks, in: *3rd European Symp. on Algorithms*, 341–354, 1995.
[28] F. Meyer auf der Heide, C. Scheideler: Communication in parallel systems, in: *SOFSEM '96*, 16–33, 1996.
[29] F. Meyer auf der Heide and B. Vöcking: A packet routing protocol for arbitrary networks, in: *12th Symp. on Theoretical Aspects of Computer Science*, 291–302, 1995.
[30] R. Ostrovsky, Y. Rabani: Universal O(congestion + dilation $+\log^{1+\epsilon} N$) local control packet switching algorithms, in: *Proc. of the 29th Ann. ACM Symp. on Theory of Computing*, 644–653, 1997.
[31] D. Peleg and E. Upfal: A tradeoff between size and efficiency for routing tables, *Journal of the ACM* **36** (1989), 510–530.
[32] C. Scheideler: *Universal routing strategies for interconnection networks,* Lecture notes in computer science, Vol. 1390, Springer-Verlag, 1998.
[33] J. P. Schmidt, A. Siegel: The spatial complexity of oblivious k-probe hash functions, *SIAM J. on Computing* **19**(5) (1990), 775–786.

[34] O. Sýkora, I. Vrťo: Edge seperators for graphs of bounded genus with applications, *Theoretical Computer Science* **112** (1993), 419–429.
[35] S. S. H. Tse, F. C. M. Lau: A lower bound for interval routing in general networks, in: *Network Journal*, June 1996.
[36] E. Upfal: An $O(\log N)$ deterministic packet routing scheme, *Journal of the ACM* **39** (1992), 55–70.
[37] L. G. Valiant: A scheme for fast parallel communication, *SIAM Journal of Computing* **11**(2) (1982), 350–361.
[38] V. V. Vazirani: A theory of alternating paths and blossoms for proving correctness of the $O(\sqrt{|V||E|})$ general graph maximum matching algorithm, *Combinatorica* **14**(1) (1994), 71–109.

Friedhelm Meyer auf der Heide

Department of Mathematics and Computer Science,
and
Heinz Nixdorf Institute
University of Paderborn
33095 Paderborn, Germany
fmadh@uni-paderhorn.de

Christian Scheideler

Department of Comuter Science
Johns Hopkins University
3400 N. Charles Street
Baltimore, MD 21218–2682
USA
scheideler@cs.jhu.edu