Alcom-FT Technical Report Series
ALCOMFT-TR-01-174

Almost Optimal Permutation Routing on Hypercubes

Berthold Vöcking*
Max-Planck-Institut für Informatik
Saarbrücken, Germany
voecking@mpi-sb.mpg.de

ABSTRACT

This paper deals with permutation routing on hypercube networks in the store-and-forward model. We introduce the first (on-line and off-line) algorithms routing any permutation on the d-dimensional hypercube in $d + o(d)$ steps. The best previously known results were $2d + o(d)$ (oblivious on-line) and $2d - 3$ (off-line). In particular, we present

- a randomized, oblivious on-line algorithm with routing time $d + O(d/\log d)$,
- a matching lower bound of $d + \Omega(d/\log d)$ for (randomized) oblivious on-line routing, and
- a deterministic, off-line algorithm with routing time $d + O(\sqrt{d \log d})$.

Previous algorithms lose a factor of two mainly because packets are first sent to intermediate destinations in order to resolve congestion. As a consequence, the maximum path length becomes $2d - o(d)$. Our algorithms use intermediate destinations as well, but we introduce a simple, elegant trick ensuring that the routing paths are not stretched too much. In fact, we achieve small congestion using paths of length at most d.

The main focus of our work, however, lies on the scheduling aspect. On one hand, we investigate well-known and practical scheduling policies for on-line routing, namely *Farthest-to-Go* and *Nearest-to-Origin*. On the other hand, we present a new off-line scheduling scheme that is based on frugal colorings of multigraphs. This scheme might be of interest for other sparse scheduling problems, too.

1. INTRODUCTION

Communication in computer networks is often performed using packet switching, that is, messages are put into packets together with a destination address and these packets are moved in store-and-forward fashion to their destinations. Typically, the problem of routing a set of packets to their destinations is broken into two parts being tackled independently: the problem of selecting the routing paths and the problem of scheduling the movements of the packets along these paths.

Let us illustrate this approach with the best known algorithms for hypercubic networks. A d-dimensional hypercube is a graph consisting of $n = 2^d$ nodes, each of which corresponding to a unique d bit binary string. Two nodes are connected by a pair of directed edges if and only if the bit strings of the nodes differ in only one bit. For simplicity, we assume time proceeds in discrete time steps. Each edge represents a communication link that can transmit a packet in every step.[1] A simple way to select the routing paths is to send messages along bit-fixing paths in dimensional order, that is, a packet is always directed along the lowest dimension in which its current position and its destination differ.

Deterministic, oblivious routing. Suppose every node appears once as the source of a packet and once as the destination of a packet. This is called *permutation routing.* Clearly, given a set of packets with specified source and destination nodes, the routing time must depend on the path selection. Let D denote the *dilation*, i.e., the length of the longest routing path, and C the *congestion*, i.e., the maximum number of routing paths using the same edge. Then $\max\{C, D\} = \Omega(C + D)$ is a trivial lower bound on the routing time. Furthermore, if the routing is *greedy* (i.e., packets only wait because other packets use the next edge on their path) then $C \cdot D$ is a trivial upper bound on the routing time. Unfortunately, using bit-fixing paths does not promise good results for permutation routing on hypercubes because these paths produce high congestion for some specific permutations. For example, if we use bit-fixing paths in order to route the bit-reversal permutation, which appears quite frequently in practical applications, then we obtain a congestion of $\sqrt{n}/d$.

Even worse, this kind of problem does not only arise when we use bit-fixing paths, it occurs for every deterministic, *oblivious* path selection strategy, i.e., for every deterministic strategy in which the routing path of a packet solely depends on the source and the destination of the packet but not on the paths of other packets. In particular, Borodin and

*Supported in part by the IST program of the EU under contract number IST-1999-14186 (ALCOM-FT).

[1] Alternatively, one might assume that nodes are connected by undirected edges. All of our results hold in this slightly more restrictive model, too. Some details in the proofs, however, need to be adapted.

STOC'01, July 6-8, 2001, Hersonissos, Crete, Greece.

Hopcroft show the following lower bound on the congestion of oblivious routing in arbitrary graphs [3]. For every deterministic, oblivious routing strategy on a network with n nodes and maximum node degree c, there is a permutation that produces a congestion of at least $\sqrt{n}/(c^{3/2})$. Kaklamanis et al. [7] sligthly improve this lower bound to $\sqrt{n}/(2c)$. Furthermore, Kaklamanis et al. give an almost matching upper bound for the hypercube. They present a deterministic, oblivious algorithm with runtime $O(\sqrt{n}/d)$. Nevertheless, we conclude that deterministic, oblivious routing leads to unacceptable worst-case results for the hypercube as it takes $\Omega(n/d)$ steps for some particular permutations.

Randomized, oblivious routing. In order to escape from this dilemma, Valiant and Brebner suggest to use *randomized routing* [20]. In randomized routing, a packet is sent from a source to a destination in two stages. The message is first sent from the source to an intermediate node selected independently, uniformly at random, and then the packet is sent from the intermediate node to the destination. Using this trick for permutation routing on hypercubes, the congestion is only $O(d/\log d)$, w.h.p.[2] The price for this small congestion, however, is that the dilation is increased by a factor of two, so that an algorithm using this trick needs at least $2d$ steps for routing arbitrary permutations. Upfal [17] was the first who introduced a packet scheduling algorithm achieving runtime $O(d)$, w.h.p, and Valiant [19] improved on this result showing how to achieve routing time $2d + O(d/\log d)$, w.h.p.

Several attempts have been made to achieve small congestion without increasing the maximum path length. For example, Valiant [19] suggests the following algorithm. For each packet starting at u and destined for v the routing algorithm chooses r nodes $w_1, \ldots, w_r$ independently, uniformly at random. Then it selects a node w_i $(1 \leq i \leq r)$ in such a way that the sum of the shortest path lengths from u to w_i and w_i to v is minimized over the r choices of w. The packet then takes bit-fixing paths first from u to w_i and then from w_i to v. Valiant shows that if $r = \sqrt{d}$ then the maximum path length of all source/destination pairs is $d + o(d)$. This trick blows up the expected congestion on every edge only by a factor of about $\sqrt{d}$.

Recently, Kim, Lehmann, and Leighton [8] obtained a similar result on the dilation increasing the congestion only by a factor of $1+o(1)$. They observed that the fraction of those intermediate destinations that lead to a path length of more than $d + \sqrt{d \log d}$ is only $o(1)$. Thus, if one chooses the intermediate destinations only from those nodes that have distance at most $d + \sqrt{d \log d}$ then the probability space is changed only slightly, so that the congestion increases only by a factor of $1 + o(1)$.

Unfortunately, however, neither Valiant nor Kim et al. can give scheduling algorithms that exploit the small dilation and congestion in order to deliver all packets in time $C+D = d + o(d)$.

Off-line routing. Until now we implicitly assumed on-line routing, that is, all routing decisions have to be made using only local control. Off-line routing allows that a global instance computes the routes of the packets and the packet schedule. The best known off-line algorithms for permutation routing on the hypercube are based on embeddings of the so-called Beneš network [2]. In this network there exist non-overlapping paths for every permutation (and one can compute these paths in time $O(n \log n)$ on a uniform RAM). Embedding the Beneš network into the hypercube yields paths with congestion at most two. The Beneš network approach basically corresponds to choosing intermediate destinations in a coordinated fashion (rather than simply at random). First, packets are routed on bit-fixing paths from the source to the intermediate destination in increasing order of dimension. Then the packets are routed from the intermediate to the final destinations using reverse bit-fixing paths in decreasing order of dimension. Unfortunately, this technique yields paths of length up to $2d$ as well.

The scheduling of the packets along the paths constructed by the Beneš network approach is straightforward. Let us consider the two collections of subpaths containing those subpaths from the sources to the intermediate destinations and those subpaths from the intermediate to the final destinations, respectively. Then the simulation of the Beneš network yields that the routing paths in each of these two collections are edge-disjoint. Thus, splitting the routing into two phases, that is, first routing all packets to the intermediate destinations and afterwards routing the packets to the final destination, there is no contention among the packets so that each phase takes time at most d. Some further improvements yield routing time $2d - 3$ [18].

Our contribution. We present a simple, elegant trick that selects paths of maximum length d while increasing the congestion only by at most a factor of two. This trick works for on-line as well as for off-line routing. In the on-line case, we simply choose a random node v and its complement $\bar{v}$ (i.e., the node with bitwise-inverse label) as alternative intermediate destinations for every packet and send the packet along the shorter of the two alternative routes. Similarly, in the off-line case, we choose the intermediate destination of a packet among a carefully chosen node and its complement. In the on-line case, complemental routing yields a congestion of $O(d/\log d)$, w.h.p. In the off-line case, it yields a congestion of at most four.

Our major contributions are the analyses of scheduling protocols on top of complemental routing. As described above, the known scheduling algorithms are not able to translate the small dilation and congestion into a small routing time. In fact, known analyses heavily rely on structural properties of the routing paths and can only show an upper bound of about $2d$ steps, even if the routing paths are much shorter. In this paper, we present the first routing algorithms with guaranteed runtime $d + o(d)$.

In Section 2, we introduce an oblivious on-line algorithm that routes any permutation in time $d + O(d/\log d)$. We achieve this routing time using a combination of the complemental routing trick with very simple, well-known and practical scheduling policies, namely Farthest-To-Go (FTG) and Nearest-To-Origin (NTO). (We want to point out that our runtime analysis goes through also for NTO/FTG on top of other path selection strategies that guarantee short paths and small congestion simultaneously, e.g., the path selection strategy suggested by Kim et al. [8].)

Furthermore, in Section 3, we give a matching lower bound for the class of oblivious on-line algorithms. We show that for any (randomized) oblivious algorithm there

[2]Throughout this paper, the term "w.h.p." means "with high probability", i.e., with probability at least $1 - n^{-\alpha}$, where $\alpha > 0$ denotes an arbitrary constant.

exists a permutation such that the algorithm needs at least $d + \Omega(d/\log d)$ steps in expectation.

Finally, in Section 4, we describe an off-line algorithm that takes only time $d + O(\sqrt{d \log d})$ using edge buffers of size three. Observe that this upper bound separates off-line and oblivious on-line routing. For off-line routing, we introduce new scheduling schemes based on frugal colorings of multigraphs. We believe that this technique is promising also for more general problems. In particular, some of the ideas might be used as a first step towards efficient algorithms for sparse, non-uniform job shop scheduling problems, i.e., job shop scheduling problems with relatively small congestion.

2. THE ON-LINE ALGORITHM

If one chooses intermediate destinations uniformly at random then the average path length is only d. When choosing n such paths, however, it will be very likely that one of them has path length almost $2d$. In the following we describe a simple and elegant trick that reduces the worst-case path length from $2d$ to d while increasing the congestion only by a factor of two. Afterwards we will describe the corresponding scheduling algorithm that achieves runtime $d + o(d)$.

2.1 Randomized complemental routing

The following algorithm is applied independently to every packet. Fix a packet p. Let s and t denote the source and destination node of this packet, respectively. We choose a d-bit binary string v uniformly at random. Now we consider two alternative routing paths, one using v and the other using the complement $\bar{v}$ as intermediate destination. For both subpaths, we use bit-fixing paths in dimensional order. If the length of the path $w = s \to v \to t$ is shorter than the length of the path $\bar{w} = s \to \bar{v} \to t$ then we choose the path w, otherwise we choose the complemental path $\bar{w}$.

The following lemma shows that this path selection strategy ensures small dilation and congestion. Fix a routing permutation π. For an edge e, let $c(e)$ denote a random variable that describes the number of routing paths that cross edge e under complemental routing. Let $c'(e)$ denote the corresponding random variable under standard randomized rounding (i.e., choosing intermediate destinations uniformly at random).

LEMMA 2.1. *Randomized complemental routing selects paths of length at most d, and* $\mathrm{E}[c(e)] \leq 2\mathrm{E}[c'(e)]$, *for every edge e.*

PROOF. First, we show that the dilation is at most d. Let $|w|$ and $|\bar{w}|$ denote the length of the paths w and $\bar{w}$, respectively, for a fixed packet p. Then

$$\begin{aligned} |w| &= \mathrm{dist}(s,v) + \mathrm{dist}(v,t) \\ &= (d - \mathrm{dist}(s,\bar{v})) + (d - \mathrm{dist}(\bar{v},t)) \\ &= 2d - (\mathrm{dist}(s,\bar{v}) + \mathrm{dist}(\bar{v},t)) \\ &= 2d - |\bar{w}| \ . \end{aligned}$$

Thus, at least one of the path has length at most d.

In order to show $\mathrm{E}[c(e)] \leq 2\mathrm{E}[c'(e)]$, we split the congestion of e under complemental routing into two parts: Let $a(e)$ denote the congestion caused by the paths using the intermediate destinations taking at first place. Clearly, $\mathrm{E}[a(e)] \leq \mathrm{E}[c'(e)]$. Now let $b(e)$ denote the congestion caused by the complemental paths. As inverting all bit-strings does not change the random experiment, $\mathrm{E}[b(e)] \leq \mathrm{E}[c'(e)]$ as well. Consequently,

$$\mathrm{E}[c(e)] = \mathrm{E}[a(e) + b(e)] = \mathrm{E}[a(e)] + E[b(e)] \leq 2\mathrm{E}[c'(e)] \ .$$

□

2.2 Packet scheduling using NTO/FTG

Suppose the packets are forwarded along routing paths chosen according to randomized complemental routing. We show that a combination of two very simple scheduling policies, namely Farthest-To-Go (FTG) and Nearest-To-Origin (NTO), delivers all packets in $d + o(d)$ steps. In case of contention for an edge, *FTG* gives precedence to packets with larger distance to their destination whereas *NTO* gives precedence to packets that are closer to their source node. Observe that FTG and NTO are equivalent if all routing paths have the same length.

In particular, we assume that packets on the bit-fixing paths to the intermediate destinations apply NTO precedence rules, packets on the bit-fixing paths to the final destinations apply FTG precedence rules, and if an *NTO packet* (i.e., a packet that has not yet reached its intermediate destination) meets an *FTG packet* (i.e., a packet that has passed the intermediate destination) then we prefer the NTO packet. Furthermore, ties are broken according to an arbitrary fixed order among the packets, e.g., using the identity number of the source nodes of the packets.

THEOREM 2.2. *Randomized complemental routing in combination with NTO/FTG yields routing time $d + O(d/\log d)$.*

The following analysis refers to specific details of the described NTO/FTG protocol, e.g., on the fact that packets on the first part of their routing paths are preferred against packets on the second part. The result itself, however, is more general. For example, there exists variations of the complemental routing technique using plain NTO or FTG scheduling for which one can show the same time bounds. We will present these results in more detail in an extended version of this paper. Here we restrict ourselves to the NTO/FTG protocol as it leads to a simpler analysis.

Before we turn to the proof itself, let us make some technical remarks on the ideas and techniques that we will use in the analysis. We will translate the NTO/FTG protocol into a protocol with priority ranks and then we will exploit these ranks in a "delay sequence" argument similar to the analyses in [9, 11, 15, 19]. The major difference to those analyses is that we assume deterministic priority ranks that increase when packets move forward rather than randomly chosen, static ranks. We have presented the idea of increasing ranks before in [14]. The so-called growing rank protocol generalizes results known for leveled networks to general networks. For the permutation routing problem on hypercubes that we consider here, however, the growing rank protocol yields only an upper bound of $O(d)$. The NTO/FTG protocol implements the idea of increasing ranks in form of a simple, deterministic scheduling policy. This variant will enable us to bound the routing time by $d + O(d/\log d)$.

Proof of Theorem 2.2. NTO and FTG can be expressed in terms of *priority ranks.* We denote the rank of a packet p while waiting for moving forward along an edge e by $r(p, e)$. If e is the ith edge ($i \geq 1$) on p's path to the intermediate destination then $r(p, e) = i - 1$. If e is the ith last edge ($i \geq 1$) on p's path to the final destination then $r(p, e) = d - i$. We can view this as if the rank of p is increased by one whenever it traverses an edge and its rank is additionally increased by $d - \ell(p) \geq 0$ when p reaches its intermediate destination, where $\ell(p)$ denotes the length of p's routing path. Furthermore, we define the *ident-rank* of p at e as $r^*(p, e) = r(p, e) + \text{source}(p)/n$. Observe that the NTO/FTG protocol ensures the following priority rule.

INVARIANT 2.3. *When an NTO (FTG) packet p delays an NTO (FTG) packet p' at a link e, then $r(p, e) \leq r(p', e)$ and $r^*(p, e) < r^*(p', e)$.*

Observe that this rule does only hold among packets of the same kind. If p is an NTO packet and p' an FTG packet, then p is preferred against p' regardless of the ranks. Nevertheless, in order to delay a packet for several time steps there needs to exist a chain of packets whose ranks satisfy certain monotonicity conditions. This is formalized in the following delay sequence argument.

DEFINITION 2.4. *For positive integers ℓ and s, let an (ℓ, s)-*delay sequence *be defined by*

A) a delay path *$\mathcal{P}$ of length ℓ, i.e., a path through the network consisting of ℓ* delay edges *$e(1), \ldots, e(\ell)$;*

B) s integers $\ell_1, \ldots, \ell_s$ with

$$1 \leq \ell_1 \leq \cdots \leq \ell_s \leq \ell \ .$$

(For simplicity of the following statements, let us additionally define $\ell_0 = 1$ and $\ell_{s+1} = \ell$.) The delay path contains at most $2d$ edges in the following order of dimensions $d, d-1, \ldots 1, d, d-1, \ldots 1$, possibly skipping some of these dimensions.

The delay sequence is called active *if there exists $s+1$* delay packets *$p_0, p_1, \ldots, p_s$ such that*

C) the routing path of packet p_i in reverse order (i.e., from destination to source) contains the subpath $e(\ell_i), \ldots, e(\ell_{i+1}) \subseteq \mathcal{P}$, for $0 \leq i \leq s$;

D) packet p_i potentially delays *packet p_{i-1} on edge $e(\ell_i)$, i.e., $r^*(p_i, e(\ell_i)) < r^*(p_{i-1}, e(\ell_i))$ or p_i is an NTO packet and p_{i-1} is an FTG packet, for $1 \leq i \leq s$.*

The proof of the following lemma contains a construction scheme of an active delay sequence that will give some insight into the components and conditions given in the definition above. Observe that the lemma limits the length of the delay path to at most d.

LEMMA 2.5. *If the routing takes T steps then there exists an active $(\ell, T - \ell)$-delay sequence with $1 \leq \ell \leq d$.*

PROOF. We give an iterative construction scheme for the delay sequence. Let p_0 be a packet that moves forward in step T (or later) along the last edge on its routing path. We follow p_0's routing path backwards to the last edge on this path where it was delayed. The packet that caused the delay is called p_1. Next we follow the path of p_1 backwards until we reach an edge at which p_1 was forced to wait by another packet that we call p_2. Now we change the packet again and follow the path of p_2 backwards, and so on.

We continue this construction until we reach a packet p_s moving along the first edge of its routing path. The path from the destination of p_s to the source of p_0 recorded by the above process is defined to be the delay path $\mathcal{P}$. Let ℓ denote the length of the delay path. For $1 \leq i \leq s$, we define the ℓ_i's in such a way that packet p_i delays packet p_{i-1} on edge $e(\ell_i)$, for $1 \leq i \leq s$. In this way, the requirements in A, B, C, and D are fulfilled.

Next let us show $\ell \leq d$. Observe that the sequence of delay packets starts with FTG packets an finishes with NTO packets. In particular, there is a unique $j \in \{0, \ldots, s-1\}$ such that

- p_i ($i \leq j$) traverses edge $e(\ell_i)$ as an FTG packet, and
- p_i ($i > j$) traverses edge $e(\ell_i)$ as an NTO packet.

Let us have a closer look at packet p_j. Recall that the path $e(\ell_{j+1}), \ldots, e(\ell_j)$ is a subpath of p_j's routing path. We distinguish two cases.

Case 1: Suppose packet p_j visits its intermediate destination on the subpath $e(\ell_{j+1}), \ldots, e(\ell_j)$. In this case, p_j passes $e(\ell_{j+1})$ as an NTO packet and arrives at $e(\ell_j)$ as an FTG packet. This implies that all FTG packets in the delay sequence are delayed only by FTG packets. In combination with Invariant 2.3, we obtain

$$r(p_i, e(\ell_i)) \ \leq \ r(p_{i-1}, e(\ell_i)) \ ,$$

for $1 \leq i \leq s$. Moreover,

$$r(p_i, e(\ell_i)) + (\ell_i - \ell_{i-1}) \ \leq \ r(p_i, e(\ell_{i-1}))$$

as ranks are increased by one (or more) whenever they traverse an edge. Iterating these two equations yields

$$r(p_s, e(\ell_s)) + (\ell_s - \ell_0) \quad \leq \quad r(p_0, e(\ell_0)) \ .$$

Furthermore, applying

$$r(p_s, e(\ell_{s+1})) + (\ell_{s+1} - \ell_s) \quad = \quad r(p_s, e(\ell_s)) \ ,$$

and using the identities $\ell_0 = 1$, $\ell_{s+1} = \ell$, gives

$$r(p_s, e(\ell)) + (\ell - 1) \quad \leq \quad r(p_0, e(1)) \ ,$$

which implies $\ell - 1 \leq d - 1$ since ranks are from $[d] = \{0, \ldots, d-1\}$.

Case 2: Suppose p_j passes edge $e(\ell_{j+1})$ already as an FTG packet. In this case, the FTG packet p_j is delayed by the NTO packet p_{j+1} on edge $e(\ell_{j+1})$. Suppose the dimension of this edge is k. Under this assumption, $\mathcal{P}$ first follows the routing paths of FTG packets in opposite direction along edges of the dimensions $d, d-1, \ldots k$ (possibly skipping some of these dimensions) and then it follows the routing paths of NTO packets along edges of the dimensions $k-1, k-2, \ldots, 1$. Consequently, the delay path has length at most d.

Finally, the equality $s = T - \ell$ can be obtained as follows. Every time step covered by our construction corresponds either to one of the s recorded delays or to a movement of a packet along one of the ℓ edges on the delay path. As we cover at least T time steps, we obtain $T \leq \ell + s$. Hence, terminating the construction at the right time yields $T = \ell + s$. $\square$

Lemma 2.5 shows that a slow packet taking T or more steps is always accompanied by an active delay sequence containing $s \geq T - d$ packets. Thus, the probability for the existence of such a delay sequence with s delay packets is an upper bound on the probability that the routing time is at least T. In the following, we will estimate this probability. First, we estimate the number of possible delay sequences. Then we multiply this number by the probability that any particular delay sequence becomes active.

Let us assign an interval of ranks to each edge. In particular, we define ℓ contiguous, non-empty *intervals of ranks* $R(1), \ldots, R(\ell)$ such that $\bigcup_i R(i) = [d]$ and the ranks in $R(i+1)$ are strictly smaller than the ranks in $R(i)$, for $1 \leq i < \ell$. Observe that the definition of an active delay sequence ensures that there exists a rank assignment $R(1), \ldots, R(\ell)$ such that $r_{e(\ell_i)}(p_i) \in R(i)$, for $0 \leq i \leq s$.

LEMMA 2.6. *For $\ell \leq d$, the number of (ℓ, s)-delay sequences with rank assignments is at most*

$$n^4 \left(\frac{\mathrm{e}(d+s)}{s}\right)^s .$$

PROOF. We calculate the number of different choices for the components of the delay sequence.

- There are n possibilities to choose the first node on the delay path and at most $2^{2d} \leq n^2$ to choose the delay edges from the at most $2d$ dimensions covered by the delay path. Observe that these choices also determines the value of ℓ.
- Furthermore, the number of possibilities to choose the sequence of non-decreasing integers $\ell_1, \ldots, \ell_s$ is
$$\binom{\ell+s}{s} \leq \binom{d+s}{s} \leq \left(\frac{\mathrm{e}(d+s)}{s}\right)^s .$$
- Finally, there are at most $2^d = n$ ways to determine the intervals of ranks.

Multiplying these numbers yields the lemma. $\square$

Now let us assume that the components of the delay sequence are fixed. We will show that the activation of the fixed delay sequence is very unlikely because of the independent random choices for the routing paths. For the activation of the delay sequence we need s delay packets that choose their intermediate destinations in such a way that all conditions are fulfilled. First, we show that the same packet cannot occur too often in the delay sequence.

LEMMA 2.7. *Each packet appears at most twice in an active delay sequence, once as NTO packet and once as FTG packet.*

PROOF. Bit-fixing paths are shortest paths. Following the arguments in [14] we will show that a packet p on a shortest routing path appears at most once in the delay sequence.

Suppose, in contrast to our claim, that packet p occurs twice in the delay sequence. Then there are i and j with $0 \leq i < j \leq s$ and $p = p_i = p_j$. Thus, the routing path of p crosses the delay path at the edges $e(\ell_j)$ and $e(\ell_i)$ in that order. Let κ denote the number of edges on the routing path of p from $e(\ell_j)$ to $e(\ell_i)$. Then the rank of p is increased κ times on this part of the routing path so that

$$r^*(p, e(\ell_i)) = r^*(p, e(\ell_j)) + \kappa . \tag{1}$$

On the other hand, each packet p_k with $i < k \leq j$ delays the packet p_{k-1} at link $e(\ell_k)$. Thus, $r^*(p_{k-1}, e(\ell_k)) > r^*(p_k, e(\ell_k))$. Furthermore, the ranks are increased on every edge on the delay path between e_j and e_i by one. Let κ' denote the number of edges on the routing path of p from $e(\ell_j)$ to $e(\ell_i)$. Then

$$\begin{aligned} r^*(p, e_i) &= r^*(p_i, e(\ell_i)) \\ &> r^*(p_j, e(\ell_j)) + \kappa' \\ &= r^*(p, e(\ell_j)) + \kappa' \\ &\geq r^*(p, e(\ell_j)) + \kappa . \end{aligned} \tag{2}$$

Note that $\kappa' \geq \kappa$, because the routing path of p is a shortest path. Clearly, (2) contradicts (1). Consequently, p appears at most once in the delay sequence. $\square$

For the time being, let us consider only NTO packets and their routing paths. For every packet p, let the binary random variable X_p be one iff packet p choose an intermediate destination in such a way that it is a *candidate for an NTO delay packet*, i.e., there exists $i \in \{1, \ldots, \ell\}$ such that the first part of the routing path of p crosses edge $e(i)$ and $r_{e(i)}(p) \in R(i)$. Define $X = \sum_p X_p$. In other words, X is the number of candidates for NTO delay packets. In the following, we will first estimate $\mathrm{E}[X]$. Afterwards, we will apply a Chernoff bound in order to obtain an upper bound on X that holds w.h.p.

LEMMA 2.8. $\mathrm{E}[X] \leq 3\sqrt{d}$.

PROOF. For the time being, let us assume that packets are routed under standard randomized routing, that is, the intermediate destinations are chosen independently, uniformly at random. (Lemma 2.1 shows that this assumption affects our upper bounds on $\mathrm{E}[X]$ only by a factor of two.)

Under the above assumption, the probability that a fixed packet has rank r after passing the first $k-1$ dimensions ($k > r \geq 0$) and actually has to traverse an edge of dimension $k > r$ is equivalent to the probability that k independent coin tosses yield $r+1$ times head. This probability is

$$\binom{k}{r+1} 2^{-k} \leq \frac{1}{\sqrt{\pi(r+1)}}$$

because $\binom{y}{x} 2^{-y} \leq \binom{2x}{x} 2^{-2x}$ and $\binom{2x}{x} \leq 1/\sqrt{\pi x}$, for integers x, y with $x \geq y \geq 1$. (The first inequality can be explained nicely as follows. For fixed x, the probability to see x heads during y coin tosses is maximized for $y = 2x$. Thus, $\binom{y}{x} 2^{-y} \leq \binom{2x}{x} 2^{-2x}$. The second inequality follows from Stirling's formulae.)

Notice that our estimation above is independent from k. In particular, the probability that a fixed packet has rank r and uses an edge of an arbitrary fixed dimension is at most $1/\sqrt{\pi(r+1)}$. Since there are n packets, the expected number of packets with rank r using edges of any fixed dimension is at most $n/\sqrt{\pi(r+1)}$. Now let us focus on the numbers of packets on individual edges. Clearly, the number of edges per dimension is n. Furthermore, the edges of any fixed dimension are indistinguishable wrt. our experiment. As a consequenc, the expected number of packets with rank r that traverse any fixed edge is at most $1/\sqrt{\pi(r+1)}$. Hence, summing over all ℓ edges on the delay path yields

$$\begin{aligned}
\mathbf{E}[X] &\leq \sum_{i=1}^{\ell} \sum_{r \in R(i)} \frac{1}{\sqrt{\pi(r+1)}} \\
&= \sum_{r=0}^{d-1} \frac{1}{\sqrt{\pi(r+1)}} \\
&\leq \int_0^d \frac{1}{\sqrt{\pi r}} dr \\
&= 2\sqrt{\frac{d}{\pi}} \leq 1.5\sqrt{d} .
\end{aligned}$$

Finally, multiplying by two yields the lemma. □

Observe that X is a sum of independent binary random variables. Therefore, X is sharply concentrated around its mean. In particular, X can be bounded above by applying a Chernoff bound as follows. Let $K \geq 1$ denote an arbitrary number. Define $a_k = \frac{K\sqrt{d}}{6 \log d}$. In this way, $\frac{Kd}{2 \log d} = 3a_K\sqrt{d}$. Lemma 2.8 shows $\mathrm{E}[X] \leq 3\sqrt{d}$. Consequently,

$$\begin{aligned}
\mathbf{Pr}\left[X \geq \frac{Kd}{2\log d}\right] &= \mathbf{Pr}[X \geq a_K(3\sqrt{d})] \\
&\leq \left(\frac{\mathrm{e}}{a_K}\right)^{a_K(3\sqrt{d})} \\
&= \left(\frac{6\mathrm{e}\log d}{K\sqrt{d}}\right)^{Kd/(2\log d)} \\
&\leq n^{-K/6} ,
\end{aligned}$$

provided that d is sufficiently large. Until now we considered only NTO packets. Analogously, let Y denote the number of candidates for the FTG delay packets. For symmetry reasons, these packets can be treated completely analogously to the NTO packets. Lemma 2.7 yields $s \leq X + Y$, and consequently,

$$\begin{aligned}
\mathbf{Pr}\left[s \geq \frac{Kd}{\log d}\right] &\leq \mathbf{Pr}\left[X \geq \frac{Kd}{2\log d}\right] + \mathbf{Pr}\left[Y \geq \frac{Kd}{2\log d}\right] \\
&\leq 2n^{-K/6} ,
\end{aligned}$$

for arbitrary $K \geq 1$. Multiplying this bound with the upper bound on the number of delay sequences given in Lemma 2.6 shows that the probability for the existence of an (s, ℓ)-delay sequence is at most

$$\begin{aligned}
&2n^{-K/6}\, n^4 \left(\frac{\mathrm{e}(d+s)}{s}\right)^s \\
&\qquad \leq 2n^{-K/6+4} \left(\frac{\mathrm{e}\log d}{K} + \mathrm{e}\right)^{Kd/\log d} ,
\end{aligned}$$

setting $s = Kd/\log d$. Finally, let α denote an arbitrary positive constant. Then this term can be bounded above by $n^{-\alpha}$, for sufficiently large constant K. This completes the proof of Theorem 2.2. □

3. A LOWER BOUND FOR OBLIVIOUS ROUTING

We show that our on-line algorithm achieves a routing time that is best possible for the class of oblivious on-line algorithms.

THEOREM 3.1. *For any (randomized) oblivious on-line algorithm $\mathcal{A}$ there is a permutation with expected routing time $d + \Omega(d/\log d)$.*

PROOF. Yao's Lemma allows us to assume that the considered algorithms are deterministic. For every deterministic algorithm $\mathcal{A}$, we will describe a random experiment selecting a permutation in such a way that, with probability $1 - o(1)$, the runtime of algorithm $\mathcal{A}$ is at least $d - \gamma + \Delta - 1$, for suitable $\gamma = O(\log d)$ and $\Delta = \Omega(d/\log d)$, whose exact values will be specified later. Thus, the expected runtime is at least $(1 - o(1))(d - \gamma + \Delta - 1) = d + \Omega(d/\log d)$, and the theorem is shown.

The permutations are selected as follows. We partition the nodes of the hypercube into $2^{d-\gamma}$ groups each of which containing those 2^γ vertices that differ only in the last γ bits. For each binary string $q \in \{0,1\}^{d-\gamma}$ let

$$\pi_q : \{0,1\}^\gamma \to \{0,1\}^\gamma$$

be a random permutation. Then for every $q \in \{0,1\}^{d-\gamma}$ and $r \in \{0,1\}^\gamma$, the packet starting at node qr is directed to node $\bar{q}\pi_q(r)$ with $\bar{q}$ denoting the complement of q.

Observe that the distance between any source and destination node in the considered class of permutations is at least $d - \gamma = d - O(\log d)$. Furthermore, observe that each source and destination pair defines a unique routing path of length at least $d - \gamma$ because we assume that $\mathcal{A}$ is deterministic. Clearly, if one of these paths has length larger than $2d$ then there exists a permutation that takes more than $2d$ steps. Consequently, we can assume in the following that the length of all possible routing paths is upper-bounded by $2d$.

An edge e is called *critical* if there are Δ routing paths each of which containing e at the same position i. The existence of a time critical edge e at position i causes a runtime of at least

$$(i-1) + \Delta + (d - \gamma - i) = d - \gamma + \Delta - 1$$

because the first packet belonging to one of the critical routing paths needs at least $i - 1$ steps to arrive at e and the last of these packets traverses e at least Δ steps later and requires at least $d - \gamma - i$ further steps to reach its destination. In the following, we will show that there exists a critical edge, with probability $1 - o(1)$.

For every node s, let $P(s)$ denote the set of those paths starting at s that can be taken by algorithm $\mathcal{A}$. Depending on the selection of the permutation, exactly one of these paths becomes a routing path. Since there are $\Gamma = 2^\gamma$ possible destinations for the packet starting at s, $|P(s)| = \Gamma$ and the probability for each path to be selected is $1/\Gamma$. Define

$$A(s,z) = \{e \,|\, e \text{ is an edge in at least } z \text{ paths from } P(s)\} .$$

The following lemma shows that there must be several edges passed by several paths from $P(s)$. We assume $\Gamma \geq dz$.

LEMMA 3.2. $|A(s,z)| \geq \Gamma/(2dz)$.

PROOF. Let B denote the edges incident on $A(s,z)$ but not in $A(s,z)$ itself. Clearly, $|B| \leq 2d|A(s,z)|$ since the node degree is d. Let $R \subset P(s)$ denote those paths whose destination is not incident on an edges from $A(s,z)$. At most $2|A(s,z)|$ paths start on nodes incident on $A(s,z)$. Therefore, $|R| \geq \Gamma - 2|A(s,z)|$. Observe that $s \notin R$ because $\Gamma \geq dz$ implies that s is incident on an edge from $A(s,z)$. Thus, all paths in R have to traverse an edge from B in order to reach their destination. As edges in B can be contained in at most $(z-1)$ paths from R, we obtain $|B|(z-1) \geq |R|$. Combining these inequalities gives

$$|A(s,z)|(z-1)d \;\geq\; \frac{|B|(z-1)}{2} \;\geq\; \frac{|R|}{2} \;\geq\; \frac{\Gamma}{2} - |A(s,z)| \ .$$

Finally, solving for $|A(s,z)|$ yields the lemma. □

Now let us define

$$T(z) \;=\; \{(s,e) \mid e \in A(s,z)\} \ .$$

Applying Lemma 3.2 yields $|T(z)| \geq n\Gamma/(2dz)$. We can interpret the elements in $T(z)$ as tokens that are placed on the edges. A token with label s on edge e means that at least z paths from $P(s)$ traverse e. Let us remove a minimal number of tokens so that the number of tokens is a multiple of z on each edge. Next we partition the tokens into groups each of which contains exactly z tokens of the same edge. Taking into account the removed tokens, the number of groups is at least

$$\frac{|T(z)| - znd}{z} \;\geq\; \frac{n\Gamma}{2dz^2} - nd \;\geq\; n \ ,$$

for $\Gamma \geq 2d^2z^2 + 2dz^2$.

Each token-group g corresponds to an edge e_g and at least z source nodes such that for source s at least z of the paths in $P(s)$ cross edge e. We define that g is *critical* if the random permutation picks at least Δ of the at least z^2 paths of g that contain e_g at the same position i, for some $1 \leq i \leq 2d$. Clearly, if g is critical than e_g is critical, too. Thus it remains to show the existence of a critical group with probability $1 - o(1)$. First, let us consider only one fixed token-group g independently of other groups.

LEMMA 3.3. $\mathbf{Pr}[g \textit{ is critical}] \geq n^{-1/2}$.

PROOF. Recall that we assume that all paths have length at most $2d$. Therefore, we can identify an integer $1 \leq i \leq 2d$ such that at least $y = \lceil z/(2d) \rceil$ source nodes represented by the token-group g have y possible routing paths containing e_g at position i. Fix a source s fulfilling this condition. Fix a set $P'(s) \subset P(s)$ of size y that contain e_g at position i. We say that s is *critical* if the random permutation picks one of the paths from $P'(s)$. Because each path in $P'(s)$ has probability $1/\Gamma$ to be selected, we obtain

$$\mathbf{Pr}[s \text{ is critical}] \;=\; \frac{y}{\Gamma}$$

Clearly, we can lower-bound the probability that at least Δ sources are critical by the probability that exactly Δ sources are critical. Suppose for a moment that all sources from group g belong to different permutation subgroups so that the path selection for different sources are stochastically independent. Then the probability that exactly Δ of the y sources of g are critical is

$$\binom{y}{\Delta}\left(\frac{y}{\Gamma}\right)^{\Delta}\left(1-\frac{y}{\Gamma}\right)^{y-\Delta}$$

Unfortunately, different sources may belong to the same permutations subgroup so that we have to deal with dependencies among the random choices for the destinations of these sources. Let E_κ denote an arbitrary event that fixes the destinations of κ source nodes different from s. W.l.o.g., we assume $\kappa \leq \Delta \leq y/2 \leq \Gamma/2$. We show that E_κ does not affect our calculations significantly. In particular,

$$\mathbf{Pr}[s \text{ is critical}|E_\kappa] \;\geq\; \frac{y-\kappa}{\Gamma} \;\geq\; \frac{y}{2\Gamma}$$

because $\kappa \leq y/2$, and

$$\mathbf{Pr}[s \text{ is not critical}|E_\kappa] \;\geq\; 1 - \frac{y}{\Gamma - \kappa} \;\geq\; 1 - \frac{2y}{\Gamma}$$

because $\kappa \leq \Gamma/2$. Hence, taking into account the dependencies, our lower bound becomes

$$\begin{aligned}\binom{y}{\Delta}\left(\frac{y}{2\Gamma}\right)^{\Delta}\left(1-\frac{2y}{\Gamma}\right)^{y-\Delta} \;&\geq\; \left(\frac{y^2}{2\Gamma\Delta}\right)^{\Delta}\left(1-\frac{2y^2}{\Gamma}\right)\\ &\geq\; \left(\frac{z^2}{8d^2\Delta\Gamma}\right)^{\Delta}\left(1-\frac{2z^2}{d^2\Gamma}\right)\end{aligned}$$

applying $\binom{a}{b} \geq \left(\frac{a}{b}\right)^b$, $(1-\frac{1}{b})^a \geq 1-\frac{a}{b}$, and $z/(2d) \leq y \leq z/d$. Finally, we set $z = 4d^2$ and $\Gamma = 64d^6$. Then the probability that there are Δ or more critical sources is at least

$$\left(\frac{16d^4}{512d^9}\right)^{\Delta}\left(1-\frac{32d^4}{64d^8}\right) \;\geq\; (2d)^{-5\Delta}2^{-1} \ .$$

Finally, we set $\Delta = \lfloor (d-2)/(10\log(2d)) \rfloor = \Theta(d/\log d)$. In this way, the probability that g is critical is at most

$$(2d)^{-5(d-2)/(10\log(2d))}2^{-1} \;=\; 2^{-d/2} \;=\; n^{-1/2} \ .$$

This completes the proof of Lemma 3.3. □

The above lemma shows that one fixed token-group is critical with probability at least $n^{-1/2}$. Recall that we have at least n token-groups. We have to show that at least one of them is critical with probability $1 - o(1)$. Two token-groups are stochastically dependent if and only if they contain source nodes of the same permutation subgroup.

One can model these dependencies by a dependency graph in which the vertices represent token-groups and two vertices are connected by an edge when the corresponding token-groups contain sources of the same permutation-subgroup. Let us determine the maximum vertex degree of the dependency graph. W.l.o.g., let us assume that Lemma 3.2 is tight so that each source node occurs in at most $\lceil \Gamma/(2dz) \rceil \leq \Gamma/z$ token groups. (Otherwise, simply remove the superfluous tokens.) Furthermore, each token-group contains at most z source nodes. Thus, the vertex degree in the dependency graph is at most Γ. Consequently, there is an independent set of at least n/Γ token-groups. The probability that at least one of the token-groups in this set is critical is at least

$$1 - \left(1 - n^{-1/2}\right)^{n/\Gamma} \;\geq\; 1 - \exp\left(\sqrt{n}/\Gamma\right) \;=\; 1 - o(1) \ .$$

This completes the proof of Theorem 3.1.

4. THE OFF-LINE ALGORITHM

4.1 Deterministic complemental routing

As suggested, e.g, in [2], we choose paths in the Beneš network and embedd this network into the hypercube. It is well-known that every permutation can be connected by non-overlapping paths in the Beneš network and that one can compute these paths in time $O(n \log n)$ on a uniform RAM. Furthermore, embedding the Beneš network into the hypercube yields paths with congestion at most two.

The above approach corresponds to choosing intermediate destinations in a coordinated fashion rather than simply at random. First, packets move on bit-fixing paths from the source to the intermediate destination in increasing order of dimension. Then packets move to their final destination using bit-fixing paths in decreasing order of dimension. Unfortunately, this approach yields routing paths of length up to $2d$.

In order to limit the maximum path length, we use exactly the same trick as in the on-line case. Fix a packet p. Let v denote the intermediate destination computed by the Beneš network algorithm. If the path via v is shorter than the path via its complement $\bar{v}$ then we choose the path via v, otherwise we choose the complemental path via $\bar{v}$.

LEMMA 4.1. *Deterministic complemental routing computes paths of length at most d with congestion at most four.*

The result on the path length follows analogously to the randomized case. Furthermore, the result on the congestion can be easily concluded from the following property of the Beneš network. Given a set of non-overlapping paths in the Beneš network, the complemental paths are non-overlapping as well. Thus, our algorithm produces two classes of non-overlapping paths in the Beneš network, each of which can be embedded with congestion at most two in the hypercube.

4.2 Off-line Scheduling using frugal colorings

Again we use a variation of the NTO or FTG policy. In the following, let P denote the set of packets, and, for $p \in P$, let W_p denote the routing path of packet p. Our algorithm computes a so-called "frugal coloring" of the packets. The properties of the coloring will be explained later. Let us now describe how we use this coloring for the scheduling.

Let B, C and γ denote suitable positive integers. (The exact values of these parameters will be described later. For the time being, you may assume $B \approx \log d$, $C \approx \sqrt{dB}$ and $\gamma \approx \sqrt{d/B}$.) Each packet $p \in P$ is assigned a color from the set $[C] = \{0..C-1\}$. The colors of the packets will be translated into priority ranks as follows.

We break each routing path W_p into γ contiguous subpaths $W_p^1, \ldots, W_p^\gamma$ in such a way that none of these subpaths contains the intermediate destination of p as internal node. Clearly, these subpaths can be chosen in such a way that the maximum length of any subpath is $2d/\gamma + 1 \le 3d/\gamma$. As the intermediate destination can only be contained as first or last node, each subpath is a bit-fixing path either in dimensional or in reverse-dimensional order. Observe that a bit-fixing path cannot join, split and join again with a routing path that is composed out of another bit-fixing path in same dimensional order and a bit-fixing path in reverse dimensional order.[3] This leads to the following observation.

OBSERVATION 4.2. *Let p and $q \neq p$ denote packets. Then the routing path W_q and the subpath $W_p^i \subseteq W_p$ $(1 \le i \le \gamma)$ can overlap at only one contiguous sequence of edges.*

We continue with the description of the priority ranks. For an edge $e \in W_p$, let $\ell_p(e)$ denote the position of e in W_p with the first edge on the path having position one. Now we define the *rank* of a packet p with color c on edge e from W_p^i by

$$r(p,e) \;=\; r(p,e,c) \;=\; \ell_p(e) + (i-1)B + c \ .$$

In other words, the initial rank of a packet is c, the rank is increased by one when the packet traverses an edge, and the rank is additionally increased by B when the packet enters the next subpath.

THEOREM 4.3. *There is a coloring that yields routing time $d + O(\sqrt{d \log d})$. The coloring can be computed in expected time $O(nd^2)$.*

PROOF. The basic idea is to translate the scheduling problem into a "frugal coloring problem". Originally, the concept of frugal coloring was introduced in [5]. They define a vertex-coloring of a graph to be *β-frugal* if for each vertex v and color c the number of times that c appears in the neighborhood of v is at most β. We introduce the following variation.

DEFINITION 4.4 (FRUGAL COLORING). *A bipartite multigraph $M = (U \cup V, E)$ with edge colors $c(e) \in [C]$ $(e \in E)$ and node colors $c(v) \in [C]$ $(v \in V)$ is colored* (C,β)-frugal *if, for each node $u \in U$, there are at most β incident edges $e = (u,v)$ with $c(v) = c(e)$.*

Suppose the edge colors are given, and the goal is to find appropriate colors for the nodes in V such that the graph is colored (C,β)-frugal. Then the edge colors can be viewed as a specification of "forbidden colors" for the nodes in V that might be violated only a few times in the neighborhood of any node in U.

The following lemma shows the existence of a frugal coloring with small C and small β, provided that parallel edges have different colors. In the following, let Δ_U and Δ_V denote the maximum degree of the vertices in U and V, respectively, and define $\Delta = \Delta_U \Delta_V$.

LEMMA 4.5. *Let $k_0 > 0$ denote a suitably large absolute constant. Suppose that all edges connecting the same pair of vertices have been assigned different colors. Then there is a (C,β)-frugal coloring of the nodes in V, provided $C \ge \Delta_U/k$ and $\beta \ge \max\{2ek, k_0 \log \Delta\}$, for any $k > 0$. Moreover, if $|V| \le 2^{\mathrm{poly}(\Delta)}$ then such a coloring can be computed in expected linear time $O(|E|)$.*

PROOF. We use Chernoff bounds and the Lovász Local Lemma (LLL) to show the existence of the described coloring. Suppose we choose the colors of all vertices in V independently, uniformly at random. Then the expected number of violations in the neighborhood of any vertex $u \in U$ is at most Δ_U/C. We define $k = \Delta_U/C$. For $h > 0$,

[3] Recall that edges are directed. In case of undirected edges these paths might overlap twice, which would require some minor modifications in the following proof.

let $p(h)$ denote the probability that there are more than $\beta = \max\{2ek, h\log\Delta\}$ violations in the neighborhood of vertex u. Observe that any violation corresponds to a different random experiment (i.e., the assignment of a color to a node in V) because redundant edges must have different colors. Therefore, applying a Chernoff bound yields

$$p(h) \leq 2^{-h\log\Delta} \leq \Delta^{-h} \quad . \tag{3}$$

Now, in terms of the LLL every node u defines a *bad event*, i.e., there are more than β violations in the neighborhood of u. The number of events that share a random experiment with any fixed event u is at most $\Delta_U(\Delta_V - 1) \leq \Delta$. Therefore, the application of the LLL require $p(h)\Delta \leq \frac{1}{4}$ We choose $h = 3$ and assume w.l.o.g. $\Delta_U \geq \Delta_V \geq 2$ then equation 3 shows that the assumptions of the LLL are fulfilled. Thus, the existence of the coloring is shown.

In order to compute the coloring, one can use an algorithmic version of the LLL presented by Molloy and Reed in [13]. The algorithms from this article, however, cannot be applied directly but need some further improvements based on techniques presented in [1, 12]. Details are explained in a full version of this paper.

□

Next we show how to translate the scheduling problem into a frugal coloring problem of a bipartite multigraph $M = (U \cup V, E)$. We define V to be the set of packets, i.e., $V = P$. Furthermore, we define

$$U \;=\; \{(W_p^i, g) \mid p \in P, 1 \leq i \leq \gamma, g \in [C]\} \quad .$$

Thus, the set U contains all combinations of subpaths and colors. It remains to specify the edges in the multigraph. We connect a vertex $u = (W_p^i, g) \in U$ with a vertex $v \in V = P$ by an edge of color c if W_v and W_p^i $(p \neq v)$ overlap at an hypercube edge e and

$$r(p, e, g) \in [r(v, e, c) - B, r(v, e, c)] \quad .$$

Intuitively, $u = (W_p^i, g)$ and v are connected by an edge of color c in the multigraph if p and v can met with close ranks on an hypercube edge in the case that p and v get assigned the initial colors g and c, respectively. The idea is that a frugal coloring minimizes the number of possible conflicts between packets with close ranks. The following lemma shows that a frugal coloring with small β leads to a small routing time.

LEMMA 4.6. *Suppose the coloring is (C, β)-frugal with $\beta \leq B$. Then the routing time is at most $d + C + \gamma B$.*

PROOF. We use the ranks of the packets as guidelines for the time steps in which the packets traverse an edge. In particular, a packet p is called *early* in step t if it wants to traverse an edge e with $t < r(p, e)$, and p is called late if $t > r(p, e) + B$. Furthermore, a packet is called *in time* if it is neither early nor late. In the following proof, we will show that all packets arrive early or in time. This yields the lemma because a packet reaches its destination in at most $C + d + \gamma B$ steps, unless it is late.

By contradiction, we assume that p is the first packet that becomes late. Suppose this happens on subpath

$$W_p^i \;=\; (e_1, e_2, \ldots, e_\delta) \quad .$$

Then p must have been arrived early or exactly in time (i.e., in step $r(p, e_1) - 1$) at the tail of edge e_1 because p would have been late on subpath W_p^{i-1}, otherwise. Thus, we can identify a subpath $W = e_\kappa, \ldots, e_\tau \subseteq W_p^i$ such that p is available in time at each of these edges and p is delayed for at least $B + 1$ times on this path. In particular, we can associate a time window $I_j = [r(p, e_j), r(p, e_j) + B]$ and an integer $t_j \geq 0$, for $\kappa \leq j \leq \tau$, such that p is delayed on e_j in I_j for t_j times and $\sum t_j \geq B + 1$.

Consider a space-time interval (e_j, I_j) $(\kappa \leq j \leq \tau)$. On one hand, packets delaying p on edge e_j in interval I_j must have a rank at least $r(p, e_j)$ as packets with smaller rank are preferred. On the other hand, the rank of these packets cannot be smaller than $r(p, e_j) - B$ because packets with smaller rank would be late if they pass edge e_j in interval I_j, which contradicts our assumption that p is the first packet that becomes late. Hence, we can restrict ourselves to a rank interval $[r(p, e_j) - B, r(p, e_j)]$, for each edge e_j $(\kappa \leq j \leq \tau)$. Now let c denote the color of p. Then the frugal edge coloring ensures that there are at most β violations in the neighborhood of the vertex $u = (W_p^i, c)$. As a consequence, there are at most β packets traversing an edge of W_p^i with ranks from the interval $[r(p, e_j, c) - B, r(p, e_j, c)]$. We summarize, the number of different packets that potentially delay packet p on path $W \subseteq W_p^i$ is at most β.

Finally, it remains to show that these packet can slowdown packet p by at most β steps. Observe that p is delayed by at most one step for every packet that enters W while p traverses this subpath. Above we have shown that there are at most β different packets that might enter W during this time. Furthermore, Observation 4.2 shows that each of these packets enter W at most once. Consequently, the maximum delay of p on W is B, which contradicts our assumptions above. Thus, Lemma 4.6 is shown. □

Next we combine the two lemmas. Let us choose a sufficiently large number of colors, that is, we set

$$C \;=\; \left\lceil \frac{2e\Delta_U}{k_0 \log\Delta} \right\rceil \;\leq\; \frac{4e\Delta_U}{k_0\log\Delta} \quad .$$

In this way, the parameter k in Lemma 4.5 can be set to $k = k_0 \log\Delta/(2e)$, so that the existence of a (C, β)-frugal coloring is guaranteed, for

$$\beta \;\geq\; \max\{2ek, k_0\log\Delta\} \;=\; k_0 \log\Delta \quad .$$

Recall that k_0 is a constant that needs to be chosen sufficiently large. W.l.o.g. the term $k_0 \log\Delta$ is integral, and we set $B = \beta = k_0 \log\Delta$. Furthermore, we define $\gamma = \lceil\sqrt{d/B}\rceil \leq 2\sqrt{d/B}$. Then Lemma 4.6 shows that the routing time is at most

$$T \;=\; d + C + \gamma B \;\leq\; d + \frac{4e\Delta_U}{k_0\log\Delta} + 2\sqrt{k_0 d \log\Delta} \quad .$$

It remains to bound Δ and Δ_U. The subpath W_p^i represented by a node $u \in U$ overlaps with at most $9d/\gamma$ paths W_v $(v \neq p)$ because the length of the subpath is at most $3d/\gamma$ and the congestion is at most four. Thus, each node $u \in U$ is adjacent to at most $9d/\gamma$ nodes in V. Since each pair of vertices in the multigraph is connected by at most $B + 1 \leq 2B$ parallel edges (of different color), we can conclude

$$\Delta_U \;\leq\; \frac{18Bd}{\gamma} \;\leq\; 18B^{3/2}d^{1/2} \;\leq\; 18(k_0\log\Delta)^{3/2}d^{1/2} \quad .$$

Similarly, the routing path represented by a node $v \in V$ overlaps with at most $3d$ subpaths and each subpath can be combined with C different colors. Thus, each node in V is adjacent to at most Cd nodes from U. Taking into account parallel edges yields $\Delta_V \leq 6BCd$. As a consequence

$$\Delta = \Delta_V \Delta_U \leq 6BCd\Delta_U \leq 24\mathrm{e}d\Delta_U^2 \leq 7777\mathrm{e}(k_0 \log \Delta)^3 d^2 .$$

We conclude that there exists a constant k_1 such that $\Delta \leq k_1 d^3$ and, hence, there exists another constant k_2 such that $\Delta_U \leq k_2 d^{1/2} (\log d)^{3/2}$. Finally, substituting, these bounds into the preceding upper bound on the routing time yields $T = d + O(\sqrt{d \log d})$. This completes the proof of Theorem 4.3.

5. CONCLUSIONS

We started with presenting a simple, elegant trick computing short routing paths with small congestion on the hypercube. Based on this trick, we introduced the first on-line and off-line scheduling algorithms for permutation routing in almost optimal time. Both scheduling protocols implement the idea that late packets are preferred against early packets. It is not clear, however, whether this idea is crucial for the results themselves or only for the analysis.

For example, consider the on-line result. On one hand, we have shown for a particular routing protocol, a combination of NTO and FTG, that it routes any permutation in time $d + O(d/\log d)$. Although the result sustains several small changes in the definition of the scheduling protocol, e.g., packets on the first part of their routing paths need not to be preferred against packets on the second part, the analysis breaks down if the priority ranks are not increased during the routing.

On the other hand, Ranade [16] (see also [9]) showed that many scheduling policies have exactly identical probabilistic behavior on random routing problems in which either the sources or the destinations of the packets are chosen independently and uniformly at random. In particular, Ranade studies *nonpredictive protocols*, i.e., algorithms placing scheduling decisions independent from the destinations. For example, FIFO, LIFO, and NTO are nonpredictive, but FTG is not. His surprising discovery is that the runtime behavior of all nonpredictive protocols on random routing problems is exactly the same.

It is not difficult to construct an example showing that the exact identity among nonpredictive protocols does not hold anymore if each node, e.g., chooses two nodes at random and sends a packet to the closest one among them. Nevertheless, in the light of Ranada's result one can imagine that many other scheduling policies achieve a performance similar to the NTO/FTG policy for complemental routing paths. Therefore, it would be interesting to know if other scheduling policies (e.g., FIFO) yield runtime $d + o(d)$ as well or why they fail.

6. REFERENCES

[1] N. Alon. A parallel algorithmic version of the Local Lemma. *Random Structures & Algorithms*, 2:367–378, 1991.

[2] F. Annexstein and M. Baumslag. A unified approach to off-line permutation routing on parallel networks. In *Proc. of the 2nd ACM Symp. on Parallel Algorithms and Architectures (SPAA)*, pages 398–406, 1990.

[3] A. Borodin and J. E. Hopcroft. Routing, merging, and sorting on parallel models of computation. *Journal of Computer and System Science*, 42:130–145, 1985.

[4] R. Cypher, F. Meyer auf der Heide, C. Scheideler, and B. Vöcking. Universal algorithms for store-and-forward and wormhole routing. In *Proc. of the 28th ACM Symp. on Theory of Computing (STOC)*, pages 356–365, 1996.

[5] H. Hind, M. Molloy, and B. Reed. Colouring graphs frugally. *Combinatorica*, to appear.

[6] T. Hagerup and C. Rüb, *A guided tour of Chernoff bounds.* Information Processing Letters 33(6):305-308, 1990.

[7] C. Kaklamanis, D. Krizanc, T. Tsantilas. Tigth Bounds for Oblivious Routing in the Hypercube. Mathematical Systems Theory 24, pp. 223-232, 1991.

[8] J. Kim, E. Lehman, and T. Leighton. Oblivious Deadlock-Free Routing in a Faulty Hypercube. In *Proc. of the 10th Symposium on Parallel and Distributed Processing*, 1999.

[9] F. T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays • Trees • Hypercubes.* Morgan Kaufmann, San Mateo, CA, 1992.

[10] F. T. Leighton, B. M. Maggs, A. G. Ranade, and S. B. Rao. Randomized routing and sorting on fixed-connection networks. *Journal of Algorithms*, 17:157–205, 1994.

[11] F. T. Leighton, B. M. Maggs, and S. B. Rao. Packet routing and job-shop scheduling in O(congestion + dilation) steps. *Combinatorica*, 14(2):167–186, 1994.

[12] F. T. Leighton, S. B. Rao, and A. Srinivasan. New algorithmic aspects of the Local Lemma with applications to routing and partitioning. In *Proc. of the 10th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 643–652, 1999.

[13] M. Molloy and B. Reed. Further algorithmic aspects of the Local Lemma. In *Proc. of the 30th ACM Symp. on Theory of Computing (STOC)*, pages 524–529, 1998.

[14] F. Meyer auf der Heide and B. Vöcking. Shortest paths routing in arbitrary networks. *Journal of Algorithms*, Vol. 31, No. 1, pp.105-131, 1999.

[15] A. G. Ranade. How to emulate shared memory. *Journal of Computer and System Science*, 42:307–326, 1991.

[16] A. G. Ranade. Equivalence of Message Scheduling Algorithms for Parallel Communication. Tech. Report YALEU/DCS/TR-512, Dept. Comp. Sci., Yale University, New Haven, CT, 1987.

[17] E. Upfal. Efficient schemes for parallel communication. *Journal of the ACM*, 31(2):507–517, 1984.

[18] X. Shen, Q. Hu, and W. Liang. Realization of an arbitrary permutation on a hypercube. *Information Processing Letters*, 51:237–243, 1994.

[19] L. G. Valiant. General purpose parallel architectures. In *J. van Leeuwen ed., Handbook of Theoretical Computer Science , Volume A*, pages 943–971, 1990.

[20] L. G. Valiant and G. J. Brebner. Universal schemes for parallel communication. In *Proc. of the 13th ACM Symp. on Theory of Computing (STOC)*, pages 263–277, 1981.