On the Benefit of Supporting Virtual Channels in Wormhole Routers

Richard J. Cole[1]
Courant Institute
New York University
New York, NY 10012
cole@cs.nyu.edu

Bruce M. Maggs[2]
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
bmm@cs.cmu.edu

Ramesh K. Sitaraman[3]
Dept. of Computer Science
University of Massachusetts
Amherst, MA 01003
ramesh@cs.umass.edu

Abstract

This paper analyzes the effect of virtual channels on the performance of wormhole routing algorithms. We show that in a network in which each edge can emulate up to q virtual channels, it is possible to route any set of b-bit messages whose paths have congestion c and dilation d in $(b+d)c(d\log d)^{1/q}2^{O(\log^*(c/d))}$ bit-steps. We also prove a nearly matching lower bound, i.e., for any values of c, d, q, and b, where $c \approx d$ and $b \geq d$, we show how to construct a network and a set of b-bit messages whose paths have congestion c and dilation d that require $\Omega(bcd^{1/q})$ bit-steps to route. These bounds imply that increasing the queuing capacity q of each edge can speed up a wormhole routing algorithm by a superlinear factor. We also present a simple randomized wormhole routing algorithm for the butterfly network. The algorithm routes a k-relation on the inputs and outputs of an N-input butterfly in $O(bq(k+\log N)(\log^{1/q} N)\log\log Nk)$ bit-steps. We present a nearly matching lower bound that holds for a broad class of algorithms.

[1]Richard Cole is supported in part by NSF Grants No. CCR-92-02900 and CCR-95-03309. [2]Bruce Maggs is supported in part by an NSF National Young Investigator Award, No. CCR-94-57766, with matching funds provided by NEC Research Institute, and by ARPA Contract F33615-93-1-1330. [3]Ramesh Sitaraman is supported in part by NSF Grant No. CCR-94-10077.

1 Introduction

Wormhole routing has become the routing method of choice in the latest generation of parallel computers, including experimental machines such as iWarp [8] and the J-Machine [32], and commercial machines such as the Intel Paragon, Cray T3D [20], and Connection Machine CM-5 [27].

In a wormhole router, the bits in a message are grouped into a sequence of *flits*, where a flit is the smallest unit of information that can be queued at a node of the network. The message is viewed as a worm that traverses the network in a bit-serial fashion. The header flit or flits, which specify the route that the message will take, proceed first. As the header flits pass through a switch, they set up a connection between the incoming channel on which they arrived, and an outgoing channel, on which they leave. The remainder of the flits contain no routing information. Instead, it is the responsibility of the switch to forward them from the incoming channel to the outgoing channel. As the header flits work their way through the network, the flits following behind become spread out across the network.

In order to keep the size of the switches small in a wormhole router, the capacity of each switch to queue flits is limited to one per incoming channel. Because a switch cannot identify which message a flit belongs to from its contents, the flits of a message must arrive on the incoming channel in a contiguous fashion. Since a switch can only queue one flit of a message, if the header flit of a message cannot advance then the flits following the header must stall.

Wormhole routing has several advantages over the other common method of routing, *store-and-forward* routing. In a store-and-forward router, a switch must store an entire message before it can forward any part of that message along the next edge on its route. Hence, a message can be viewed as making discrete hops, called *message steps* from switch to switch as it traverses its path. If a b-bit message traverses a path of length d, and is never delayed, then it will reach its destination in bd bit steps (assuming each channel can transmit one bit in each bit step). In a wormhole router, however, the header flit does not wait for the rest of the message. The first bit of the header flit arrives at its destination after d steps, and the last bit of the message arrives after $d + b - 1$ steps. The difference in time is due to the better utilization of network edges by the wormhole router; in a wormhole router, multiple edges of the network can simultaneously transmit flits belonging to a single message, whereas in the store-and-forward algorithm only one edge can transmit a flit of the messsage at any time. In addition to reduced latency, wormhole routing also has the advantage that it can be implemented with small, fast switches.

Wormhole routing owes much of its recent popularity to an influential paper by Dally and Seitz [13], which introduced the method. Much of the paper by Dally and Seitz is devoted to the design of wormhole routing algorithms that avoid *deadlock*. A wormhole routing algorithm can deadlock if the header flit of every worm is prevented from moving because the queue that it wants to enter is full. For example, a pair of worms can deadlock if the head of the first worm wants to enter a queue

occupied by the tail of the second worm, and vice versa. The solution to this problem in the paper by Dally and Seitz is to allow each physical channel to emulate several virtual channels, and to construct a virtual network in which the worms cannot form cycles. The virtual channels share the physical wire (or wires) provided by the physical channel, but the switch maintains a separate queue for each virtual channel. This solution has also been implemented in hardware. For example, each physical channel of the iWarp machine supports 4 virtual channels.

This paper provides a rigorous and quantitative analysis of the impact of virtual channels on the speed of a wormhole router. We make no queuing-theory assumptions about the independence of the behavior of the switches in the network. We present two types of results: those that hold for networks with arbitrary topologies, and those that hold for butterfly networks. In both cases, we show that allowing each physical channel to support q virtual channels can yield a speed up that is larger than linear in q.

1.1 Congestion, dilation, queue size, and message length

Throughout this paper we express the performance of routing algorithms in terms of four parameters: congestion, dilation, queue size, and message length.

The congestion and dilation of a set of messages are properties of the paths taken by those messages. We will generally decouple the process of selecting paths for the messages from the process of scheduling the movements of the messages along their paths. The *congestion* c of a set of messages is the maximum number of messages that traverse any edge of the network. The *dilation* d is the length of the longest path taken by a message.

The message length b is simply the number of bits in the message, including any header bits required to route the message through the network.

The queue size q is the maximum number of different messages that can be queued, in whole or in part, at the head of any network edge, i.e., q is the maximum number of virtual channels that can be supported by any physical channel. Throughout this paper, we assume that after a flit traverses a network edge, it is queued at the head of that edge. We also assume that initially each message is stored in a queue at its node of origin, and ultimately is stored in a queue at its desination.

1.2 The butterfly network

Some of the results in this appear apply to a popular type of multistage interconnection network called a *butterfly network*. Figure 1 shows an 8-input butterfly network. Throughout this paper, we use the following terminology to describe butterfly networks. An N-input butterfly has $N(\log N + 1)$ nodes arranged in $\log N + 1$ *levels* of N nodes each. (Throughout this paper we use $\log N$ to denote $\log_2 N$.) Each node has a distinct label $\langle w, i \rangle$ where i is the level of the node ($0 \leq i \leq \log N$)

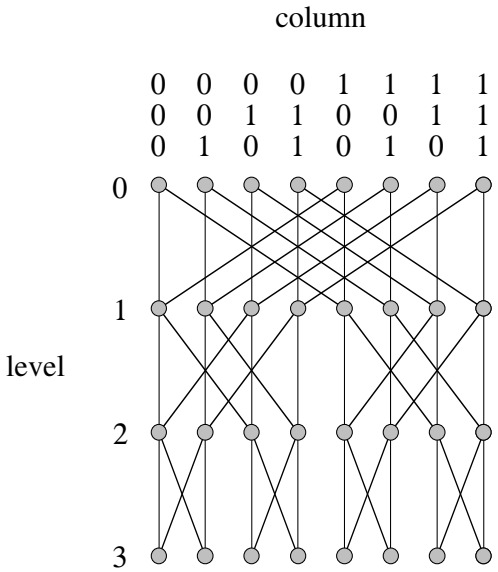

Figure 1: An 8-input butterfly network.

and w is a $\log N$-bit binary number that denotes the *column* of the node. All nodes of the form $\langle w, i\rangle$, $0 \leq i \leq \log N$, are said to belong to column w. Two nodes $\langle w, i\rangle$ and $\langle w', i'\rangle$ are linked by an edge if $i' = i + 1$ and either w and w' are identical or w and w' differ only in the bit in position i'. (The bit positions are numbered 1 through $\log N$.) We call the first type of edge a *straight edge* and the second a *cross edge*. The nodes on level 0 are called the *inputs* of the network, and the nodes on level $\log N$ are called the *outputs*. Level 0 is the *top* of the butterfly, and level $\log N$ is the *bottom*. Sometimes the level 0 and $\log N$ nodes in each column are assumed to be the same node. In this case, the butterfly is said to *wrap around*.

We study two canonical routing problems on butterfly networks: the *k-relation routing problem* and the *random routing problem* with k messages per input. In a k-relation routing problem at most k messages originate at each input and aty most k messages are destined for each output. The special case of $k = 1$ is called *permutation* routing. In a random routing problem with k messages per input, each of the k messages at each of the inputs selects independently a random output as its destination.

1.3 Previous and related research

There is a vast literature devoted to the subject of routing messages in networks. For a broader treatment of the subject, see the survey paper by Leighton [25], or

his book [22]. In what follows we describe only the research most closely related to this paper.

Network independent approaches The idea of decoupling the path selection process from the scheduling process, and then analyzing the scheduling process alone, was first used by Leighton, Maggs, and Rao [24]. They proved that for any set of messages whose paths are edge-simple (i.e., no path uses the same edge more than once) and have congestion c and dilation d, there is a store-and-forward schedule that routes the messages in $O(c+d)$ message steps and never stores more than a constant number of messages in the queue at the head of an edge. The $O(c+d)$ bound improves on the naive $O(cd)$ bound, and is optimal. Translated to bit steps, the bound is $O(b(c+d))$, which is optimal for $c > d$ (because some edge must transmit bc bits). Note that there is no restriction on the structure of the network, and that the size of the network and number of messages do not appear in the running time. The proof used the Lovász Local Lemma and was nonconstructive.

For the special case of leveled networks, Leighton, Maggs, Ranade, and Rao [23] presented a simple on-line store-and-forward algorithm for routing any set of N messages in a leveled network with depth L in $O(c+L+\log N)$ message steps. In a *leveled* network with depth L, each node is labeled with an integer between 0 and L, and each edge with its tail on level i, $0 \leq i < L$, has its head on level $i+1$.

Mansour and Patt-Shamir [29] then showed that if messages are routed greedily on shortest paths, then all of the messages reach their destinations within $d+N-1$ steps, where N is the total number of messages. These schedules may be much longer than optimal, however, because N may be much larger than c.

Meyer auf der Heide and Vöcking [31] later devised a simple on-line randomized algorithm that routes all messages to their destinations in $O(c+d+\log N)$ steps, with high probability, provided that the paths taken by the messages are short-cut free (e.g., shortest paths).

Recently, Leighton and Maggs [26] devised an algorithm for finding a store-and-forward routing schedule of length $O(c+d)$ on any network. The algorithm is based on the techniques of Beck [5] and Alon [2] for making the Lovász Local Lemma constructive. It uses information about the entire network and all of the messages, and runs in $o(P \log P)$ time, where P is the sum of the lengths of the paths taken by the messages.

In a classic paper, Borodin and Hopcroft [9] proved that any deterministic oblivious store-and-forward algorithm must take a least $\Omega(\sqrt{N/d^3})$ message steps to route some permutation on any N-node degree d network. An *oblivious* routing algoritm is one in which every message's path is determined solely by its origin and destination, and not by the actions taken by other messages. The Borodin-Hopcroft lower bound was later improved to $\Omega(\sqrt{N}/d)$ by Kaklamanis, Krizanc, and Tsantilas [18]. These bounds can be translated to lower bounds (in bit steps) for oblivious wormhole routing algorithms simply by multiplying the time by the message length, e.g.,

the Kaklamanis et al. bounds becomes $\Omega(b\sqrt{N}/d)$ bit steps. Both of these bounds hold independent of the capacity of nodes and edges to queue messages.

The Borodin-Hopcroft lower bound was extended to randomized oblivious algorithms by Aiello, Leighton, Maggs and Newman [1], who proved that on any degree-d network, almost all permutations require $\Omega(\log N/(\log d + \log\log N))$ message steps, or $\Omega(b \log N/(\log d + \log\log N)$ bit steps. (Note that in this paper we include the header bits in the length, b, of a message, whereas in [1] these bits are not included, and instead an explicit $\log_d N$ term is added.) For constant-degree networks such as the butterfly, this bound is $\Omega(b \log N/\log\log N)$ bit steps. Borodin, Raghavan, Schieber, and Upfal [10] later showed that for $d \leq n/\log^3 n$, any oblivious randomized single-port permutation routing algorithm requires $\Omega(\log_d N + \log N/\log\log N)$ message steps, on average. Their bound is stronger when d is large.

For wormhole routing, Ranade, Schleimer, and Wilkerson [34] showed how to construct a network with queue size $q = 1$, and a set of b-bit messages whose paths have congestion c and dilation d in which the optimal wormhole routing schedule requires $\Omega(bcd)$ time. Ranade et al. also showed that on any network, any set of b-bit messages whose paths have congestion c and dilation d can be routed in $O(bcd)$ bit steps. The $O(bcd)$ bound improves on the naive $O((b+d)cd)$ bound. The result of Leighton, Maggs, and Rao implies that a store-and-forward algorithm can route the same messages in $O(b(c+d))$ time. Hence, in this example store-and-forward routing is more efficient than wormhole routing.

Routing on butterfly networks A large number of store-and-forward algorithms have been designed for butterfly networks. Since this paper presents no new results in this area, we will not review the previous results. Descriptions of several of the algorithms can be found in [22, 25]. Instead, we focus on algorithms for wormhole routing.

In two early papers, Beizer [6] and Beneš [7] showed that it is possible to route edge-disjoint paths between the inputs and outputs of a Beneš network in any permutation. A Beneš network is simply two back-to-back butterfly networks. Waksman [38] gave an elegant algorithm for determining how the nodes should be set in order to realize any particular permutation. Waksman's algorithm can be used for wormhole routing. It shows how to route any permutation of N b-bit messages in $O(b + \log N)$ steps, or any k-relation in $O(kb + \log N)$ steps. This algorithm was implemented on the IBM GF-11 parallel computer. Waksman's algorithm, however, uses global knowledge of the permutation in order to set the switches of the network.

Kruskal and Snir [21] showed that if each input in an N-input butterfly network sends a message to a randomly-chosen output, and at most one message can use any edge of the network, then the expected number of messages that reach their destinations is $\Theta(N/\log N)$.

Koch [19] generalized the result of Kruskal and Snir by showing that if each edge can support q messages, then the expected fraction of messages that get through

is $\Theta(N/\log^{1/q} N)$. Thus, Koch observed, as we do in this paper, that increasing the capacity of each edge by a constant factor may increase the performance of a wormhole routing algorithm by more than a constant factor. Koch's work was motivated by the fact that on the BBN Butterfly parallel computer [4], $q > 1$.

Maggs and Sitaraman [28] generalized the previous two results by showing that, by making two passes through a butterfly, it is possible to route a $\Theta(N/\log^{1/q} N)$ fraction of any permutation (rather than only a random permutation), with high probability.

In Problem 3.285 of [22], Leighton describes an algorithm for solving a random routing problem in which each input has one b-bit message to send. The algorithm runs in $O((b+\log N)\log N)$ bit steps. In Problem 3.286, he shows that the algorithm can be converted to one that routes any permutation using Valiant's idea [37] of first routing to random intermediate destinations. For the interesting case of $b = O(\log N)$, the time is $O(\log^2 N)$ bit steps. The algorithm can easily be generalized to the case $k > 1$, and runs in $O(k(b + \log N)\log N)$ bit steps. For the interesting case of $b = O(\log N)$ and $k = \log N$, the time is $O(\log^3 N)$ bit steps.

Felperin et al. [17] independently discovered an $O(\log^4 N)$ bit-step algorithm for solving a random problem for the case $b = O(\log N)$ and $k = \log N$, and then Ranade et al. [34] discovered an $O(\log^3 N \log\log N)$ bit-step algorithm.

Ranade et al. also proved an $\Omega(\log^3 N/(\log\log N)^2)$ lower bound on the number of bit steps required for a wormhole routing algorithm to route a $\log N$-relation when $b = \log N$ and $q = 1$. This lower bound is nearly tight.

Other wormhole routing algorithms Following the paper by Dally and Seitz, many papers were written on the subject of wormhole routing. Most of these papers can be placed into one of the following four categories:

1. papers that describe implementations of wormhole routing switches,
2. papers that make queuing theory independence assumptions to analyze the performance of wormhole routing algorithms, and validate the analysis via simulations,
3. papers that describe deadlock-free wormhole routing algorithms, and
4. papers that prove rigorous worst-case bounds on the running times of wormhole routing algorithms.

In the first category Seitz et al. describe the architecture of the Ametek Series 2010 Multicomputer [35], and Dally and Seitz [16] describe the Torus Routing Chip.

In the second category, two of the most influential papers were written by Dally [14, 15]. The first analyzes the behavior of wormhole routing algorithms for k-ary n-cubes. The second analyzes the effect of virtual channels on the throughput of multistage networks.

In the third category, several types of deadlock-free algorithms have been devised. A *minimal* algorithm is one in which messages travel only along shortest paths. An *adaptive* routing algorithm is one in which the path taken by a message may depend on the actions of the other messages in the network. (In contrast to an oblivious routing algorithm.) A *fully-adaptive minimal* algorithm is one that considers every possible shortest path for a message.. Minimal deadlock-free algorithms have been designed for de Bruijn and shuffle exchange networks [11]. Fully-adaptive minimal deadlock-free algorithms have been devised for trees [30], meshes [33], toruses [12], and hypercubes [33]

In the last category, wormhole routing algorithms have been designed for hypercubes, multibutterflies, trees, and meshes with constant dimension. (A mesh with constant dimension is a k-ary n-cube where n is a constant.)

On the hypercube network, Aiello, Leighton, Maggs, and Newman [1] devised a randomized algorithm for routing N b-bit messages on an N-node hypercube in any permutation in $O(b + \log N)$ bit steps. The algorithm requires q to be a small constant larger than one, and assumes that each hypercube node can service all $\log N$ of its edges simultaneously.

On the multibutterfly network, Arora, Leighton, and Maggs [3] devised an algorithm for routing N b-bit messages from the inputs to the outputs of an N-input network in $O(b + \log N)$ bit steps. The algorithm can also be applied to a multi-Beneš network (two back-to-back multibutterfly networks), which is a nonblocking network.

On trees and meshes with constant dimension, Ranade et al. [34] presented off-line algorithms for routing in $O(bc + d)$ steps. The lengths of these algorithms are optimal.

1.4 Our results

Our most general result is a proof that if each edge can queue up to q message flits, then it is possible to route any set of b-bit messages whose paths are edge simple and have congestion c and dilation d in $(b+d)c(d \log d)^{1/q} 2^{O(\log^*(c/d))}$ bit-steps. The proof is nonconstructive, but can be made constructive using the techniques in [26]. The algorithm uses information about the entire network and all of the messages in order to find a routing schedule. Hence, it is not an on-line algorithm. We also prove a nearly matching lower bound, i.e., for any values of c, d, q, and b, where $c \approx d$ and $b \geq d$, we show how to construct a network and a set of paths with congestion c and dilation d and a set of b-bit messages that require $\Omega(bcd^{1/q})$ bit-steps to route. The proof of our lower bound is similar to the $\Omega(bcd)$ lower bound proved by Ranade et al. for the case $q = 1$. These matching upper and lower bounds imply that increasing the queueing capacity of each edge can speed up a wormhole routing algorithm by a superlinear factor, an observation that was first made by Koch [19] for circuit-switching on the butterfly.

For the butterfly, we present a simple randomized wormhole routing algorithm.

The algorithm routes a k-relation on the inputs and outputs of an N-input butterfly in $O(bq(k+\log N)(\log^{1/q} N)\log\log Nk)$ bit-steps. The algorithm has the same performance (to within small factors) as the fastest previously known algorithms when $q = 1$, but is faster by a factor of about $\log^{1-1/q} N$ for $q > 1$. We present a nearly matching lower bound that holds for a broad class of algorithms. Our algorithm, however, does not fall into this class.

For the case $b = logN$, $k = \log N$, and $q = 1$, the lower bound of Ranade et. al is $\Omega(\log^3 N/(\log\log N)^2)$, but several known store-and-forward routing algorithms run in $O(\log^2 N)$ bit steps (see Leighton[22]). Hence, the store-and-forward algorithms are faster. The switches executing the store-and-forward algorithms, however, must each have the capacity to queue an entire message, i.e., $O(\log N)$ bits. For $q = \log\log N$, the time for our wormhole routing algorithm drops to $O(\log^2 N(\log\log N)^2)$. Hence, a wormhole routing algorithm can achieve nearly the same time bound as the store-and-forward algorithm, while queuing at most $O(\log\log N)$ bits at the end of each edge.

1.5 Outline

The remainder of this paper is organized as follows. Section 2 presents the results for general networks. Section 3 presents the results for butterfly networks.

2 Network independent analysis

In Section 2.1 we show that for any set of b-bit messages whose paths have congestion c and dilation d in any network, there is a schedule for routing the messages in $(b+d)c(d\log d)^{1/q}2^{O(\log^*(c/d))}$ bit steps.

In Section 2.2 we show that there exists a routing problem that requires at least $\Omega(bcd^{1/q})$ bit steps to route.

2.1 Upper bounds on schedule length

The method used is to partition the messages by coloring them. Successive collections of messages of a given color are released at times separated by time intervals of length $\max\{b,d\}$. The messages of a given color are selected so that at most q messages use any given edge. This provides a non-blocking schedule given q-multiplexing.

The coloring is obtained by repeated refinement. The following definition will be helpful.

Definition 2.1 *The* multiplex size, *m, of a color class is the maximum over all edges of the number of messages in the color class crossing that edge.*

Initially, all messages are placed in a single color class. So the class has multiplex size c.

First, suppose $c \leq d$. In this case, a two step construction is used. In Step 1, the multiplex size of the color classes is reduced to $\log d$ by creating $O(c/\log d)$ color classes. In Step 2, the multiplex size is reduced to q, by replacing each color class with $O(\log d(d \log d)^{1/q})$ color classes. Overall, this gives a schedule of length $O((b+d)c(d \log d)^{1/q})$.

Each step of the construction uses the Lovász Local Lemma (LLL) [36, pp. 57–58]. The Lemma relates the probability p_b of a "bad" event, where events have limited dependence, δ, to the existence of a scenario with no bad events, a *good* scenario; LLL shows there is a good scenario if $4\delta \cdot p_b < 1$.

Let m_s bound the multiplex size of the color class being refined, and let m_f be the bound on the desired multiplex size for the refined color classes. Then the bad event is that thre are m_f messages assigned the same color as a given message M which all use the same edge e of the network. The dependence is at most $m_s \cdot d$ ($\leq m_s$ messages each traversing $\leq d$ edges). The good scenario is the existence of the refined color classes.

For $c > d$, we apply the refinement strategy $O(\log^*(c/d)$ times. To reduce the multiplex size of the color classes to d requires the creation of $O(2^{O(\log^*(c/d)})$ color classes. In one refinement step, the bound on the multiplex size of the color classes is reduced from x to $\max\{d, \log x\}$, for arbitrary $x > d$.

Theorem 2.2 *Given a routing with parameters b, c, d, there is a q-multiplexing schedule of length*

1. $O((b+d)c(d \cdot c)^{1/q})$, *for* $c < \log d$,
2. $O((b+d)c(d \log d)^{1/q})$, *for* $\log d \leq c \leq d$, *and*
3. $O((b+d)c(d \log d)^{1/q} 2^{O(\log^*(c/d)})$, *for* $c > d$.

Proof: We consider a color refinement step. Let m_s be the multiplex size of a color class being refined and let m_f be the bound on the desired size of the refined color classes. Let r be the number of refined color classes. Let M be a message traversing edge e. The probability that there are m_f other messages in the same refined class as M traversing edge e is bounded by:

$$p_b = \left(\begin{array}{c} m_s \\ m_f \end{array} \right) 1/r^{m_f}$$

The dependence is $\delta = m_s \cdot d$. If $4p_b \cdot \delta < 1$, then there exists a collection of refined color classes with at most m_f messages of a given color traversing each edge.

Case 1: $m_s = x$, $m_f = \max\{d, \log x\} >= d$. Choose $r = 32x/\max\{d, \log x\}$. Then $4p_b \cdot \delta < 1$.

Case 2: $d \geq m_s > \log d$, $m_f = \log d$. Choose $r = 32m_s/\log d$. Then $4p_b \cdot \delta < 1$.

Case 3: $m_s \leq \log d$, $m_f = q$. Choose $r = 8(d \cdot m_s)^{1/q}(\log d)/q$. Then $4p_b \cdot \delta < 1$.

Over all the levels of refinement, the number of color classes created is

$$32^{\max\{0,2+\log^*(c/d)\}} c/q(d \cdot \min\{c, \log d\})^{1/q}.$$

Each color class is allocated $b + d$ time for its traversal of the network, which suffices as no message is blocked.

□

For $b < d$, we would like to obtain a schedule of length say $O(bc(d \log d)^{1/q})$ or even $O(bc(b \log b)^{1/q})$, as the latter would interpolate gracefully with the $O(c + d)$ bound for the case $b = 1$. We have not achieved this bound, but we can improve significantly on the bound of Theorem 2.2.

We use a somewhat different coloring and scheduling strategy. We note that for two messages M and M' both using edge e, given the schedule for M, there are exactly $2b - 1$ time slots in which M' can interfere with M on edge e. This leads us to the strategy of releasing a color class every $2b$ steps. Then our goal is to color messages so that for each message M and edge e that M traverses, there are fewer than q messages colliding with M on edge e. With sufficient colors, we can again assure there exists a coloring using LLL.

Unfortunately, it is not clear how to apply this strategy iteratively. Instead, we use the previous refinement strategy, creating color classes of size s for an appropriate $s < d$ dependent on b. The messages in each color class will be released in time intervals separated by gaps of length d to ensure they do not interfere. For each color class, the messages are recolored according to the strategy described in the preceding paragraph. This leads to the following bounds.(For brevity, we report the results only for $c \leq d$, but they generalize exactly as in Thereom 2.2.)

Theorem 2.3 *Given a routing with parameters b, c, d, $b < d$, $c \leq d$, there is a q-multiplexing schedule of length*

1. $O(bc(d^2/b)^{1/(q+1)})$, *for* $b \leq d^{(q-1)/q}/(\log d)^{(q+1)/q}$, *and*
2. $O(bc(d \log d)^{1/q})$, *for* $b \geq d^{(q-1)/q}/(\log d)^{(q+1)/q}$.

2.2 Lower bounds on schedule length

We show that there exists a routing problem that requires at least $\Omega(bcd^{1/q})$ time to route. This matches our upper bounds within small factors. This proof is a generalization of the proof for $q = 1$ in [34]. The structure of our proof is similar to theirs.

Theorem 2.4 *There exists a network and a set of $M = p^{q+1}$ messages of length $b > 2p^q + 1$ such that the total time required to route all the messages is $\Omega(bcd^{1/q})$, where p is any prime number.*

Proof: The key idea is to construct a set of messages such that *every* set of $q+1$ messages passes through a specific edge. This implies that if the messages are long enough, at most q messages can move at any time step. This gives the required bound.

The network is constructed using two types of edges: primary edges and secondary edges. First, we include p^q primary edges such that no two primary edges share an end-point. Next, we connect the end-point of each primary edge to the end-point of every other primary edge using secondary edges.

The messages and their paths through the network is constructed as follows. Associate with each of the $M = p^{q+1}$ messages a distinct $q+1$-tuple $\langle m_1, m_2, \cdots, m_{q+1} \rangle$, where $m_i \in GF(p)$. Each primary edge is associated with a linear equation of $q+1$ variables of the form $a_1x_1 + a_2x_2 + \cdots + a_{q+1}x_{q+1}$ such that the co-efficients of the equation $a_i \in GF(p)$. Clearly, there are p^{q+1} such primary edges since there are that many distinct values for $\langle a_1, a_2 \cdots, a_{q+1} \rangle$. A message $\langle m_1, m_2, \cdots, m_{q+1} \rangle$ passes through a primary edge $\langle a_1, a_2 \cdots, a_{q+1} \rangle$ iff

$$a_1m_1 + a_2m_2 + \cdots + a_{q+1}m_{q+1} = 0 \tag{1}$$

Each message uses primary and secondary edges alternately till it reaches its destination. Each message originates at the end-point of one of its primary edges. It uses this primary edge and then uses a secondary edge to reach its next primary edge, until it has traversed its last primary edge.

We now evaluate the congestion c and dilation d of the routing problem. There are exactly p^q messages using any primary edge, since for any fixed primary edge $\langle a_1, a_2 \cdots, a_{q+1} \rangle$ there are at most p^q values of $\langle m_1, m_2, \cdots, m_{q+1} \rangle$ that satisfy equation 1. It is easy to see that the congestion of a primary edge is more than the congestion of a secondary edge. Therefore, congestion $c = p^q$. Further, each message passes through exactly p^q primary edges, since for any fixed message $\langle m_1, m_2, \cdots, m_{q+1} \rangle$ there are at most p^q values of $\langle a_1, a_2 \cdots, a_{q+1} \rangle$ that satisfy equation 1. Thus the dilation d, including the secondary edges, is $2p^q + 1$. We choose the length of each message b to more than the dilation.

We now evaluate the time to route the messages. Using an argument similar to Theorem 3.12, we can show that the routing proceeds in phases with the messages in the i^{th} phase arriving at time $d + ib$. Further, no set of $q+1$ messages that belong to a phase can pass through an edge. Since our construction ensures that every set of $q+1$ pass through an edge, each phase can contain no more than q messages. Therefore, to route a total of $p^{q+1} = \Omega(cd^{\frac{1}{q}})$ messages, at least $\Omega(cd^{1/q}/q)$ phases are required. Now, since each phase takes at least b time we need at least $\Omega(bcd^{1/q})$ time to route all the messages. □

3 Routing on butterfly networks

In Section 3.1, we provide a wormhole routing algorithm for the butterfly network. We show that an arbitrary k-relation can be routed on an N-input butterfly in time

$O(bq(k+\log N)(\log^{1/q} N)w_1(N,k))$ steps, where $w_1(N,k)$ is a slowly-growing function of N and k. As a typical example, when $b = k = \Theta(\log N)$, the algorithm routes all the messages in time $O(q(\log^{2+1/q} N)w_1(N,k))$, where $w_1(N,k) = O(\log\log N)$. This matches the best known results for $q = 1$ (to within the $w_1(N,k)$ factor), and is faster by a factor of about $\log^{1-1/q} N$ for $q > 1$. The results in Section 3.1 also apply to the random routing problem with k messages per input.

In Section 3.2, we provide a lower bound for a broad class of algorithms that matches the above upper bound to within small factors. Specifically, we show that a random routing problem with k messages per input requires $\Omega(bkl^{1/q}/w_2(N,k))$, where $w_2(N,k)$ is a slowly-growing function of N and k. As a typical example, when $b = k = O(\log N)$, the the lower bound evaluates to $\Omega(\log^{2+1/q} N/w_2(N,k))$, where $w_2(N,k) = O(\log\log N)$. Thus this matches the upper bound within a small polyloglog factor. The results in Section 3.2 also apply to routing a random k-relation.

3.1 Butterfly algorithm

The algorithm for routing a k-relation on an N-input butterfly network proceeds in rounds. There will be a total of $2\log\log(Nk) + 1$ rounds. Each round consists of the following steps (R takes on values from 0 to $2\log\log(Nk)$ and denotes the round number).

1. Each input makes two identical copies of each undelivered message that it holds. (Skip this step for round 0).

2. Each message picks a color independently and randomly from the set of colors $\{1, \cdots, \Delta\}$, where $\Delta = \beta k \log^{\frac{1}{q}} N$, for some constant β.

3. In each round, we run Δ sub-rounds – in each sub-round we route messages of a particular color. Each message makes two passes through the butterfly. In the first pass, the message routes from its input node to a random intermediate node in level $\log N$ of the butterfly. In the second pass, the message routes from the random intermediate node to its actual destination output (See Figure 3.1).

4. During the process of routing, if a message is delayed at a switch, then the message is discarded. Each such undelivered message will be resent in the next round.

We prove the following bound on the total time taken for a copy of each of the kN messages to reach their respective destinations.

Theorem 3.1 *The above algorithm delivers a copy of each message in a k-relation routing problem in $O(b(k+\log N)(\log^{1/q} N)w_1(N,k))$ bit-steps, with high probability, where b is the number of bits per message, q is the number of virtual channels, and $w_1(N,k) = \log\log(Nk)$.*

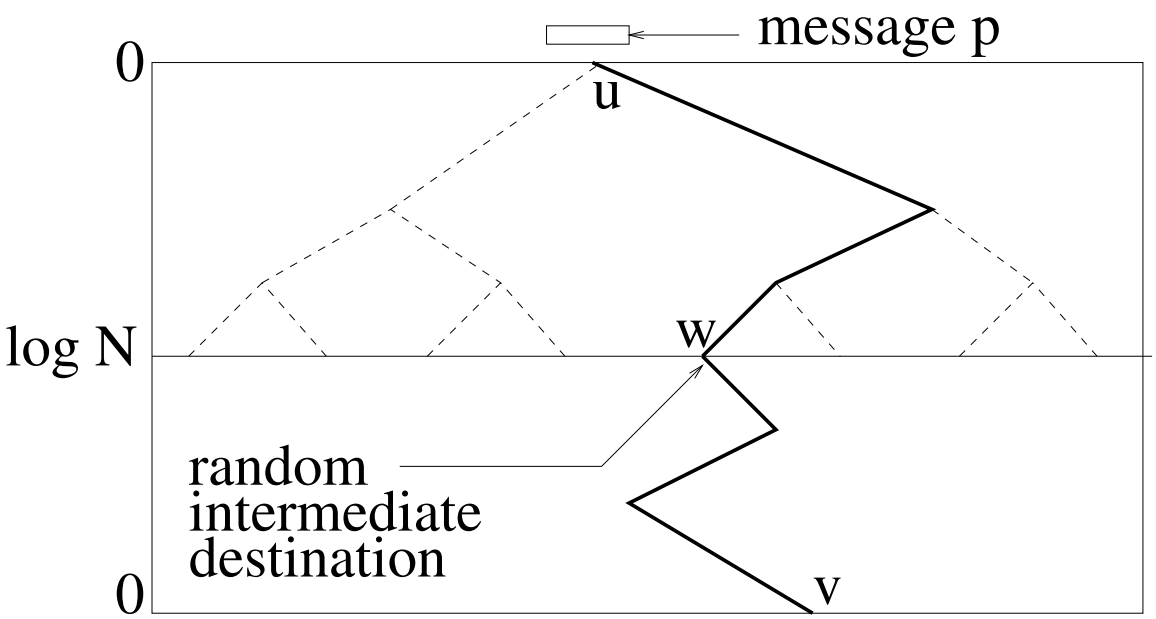

Figure 2: Routing message p using two passes through the butterfly

We focus on the case where $k \geq \log N$. The case when k is smaller follows. We show that the following invariant is maintained.

Invariant 3.2 *After Step 1 of each round, the total number of messages originating in a specific input or destined for a specific output is at most k, with high probability.*

We show that Invariant 3.2 holds by using induction on the number of rounds. Clearly, Invariant 3.2 holds at the beginning of round 0, since each input has k messages and each output is the destination for k messages. Using the following lemmas, we show in Theorem 3.6 that if the Invariant 3.2 holds in round i, it continues to hold in round $i+1$.

Lemma 3.3 *Let $m_1, m_2, \cdots, m_t$ be a set of t messages such that no two messages in this set pick the same color in Step 2 of a certain round. The probability that no message in this set is delivered in that round is at most $1/2^{12t}$.*

Proof: Let the $P_1, P_2, \cdots, P_t$ denote the paths taken by the messages $m_1, m_2, \cdots, m_t$ respectively. Further, let $c_1, c_2, \cdots, c_t$ denote the distinct colors picked by these messages. Note that m_i is delayed only if there are q other messages of the same color using some edge on its path. Thus, if the algorithm fails to deliver all t messages, then there must exist in each path P_i an edge e_i such that the q other messages of color c_i use this edge.

We bound the required probability by enumerating the different configurations that can result in the failure of all t of the messages. With each path P_i associate an edge e_i that spans levels l_i and $l_i + 1$ of the butterfly. The probability that q messages of color c_i pass through edge e_i is evaluated as follows. If edge e_i is in the first pass (resp. second pass) then there are 2^{l_i} inputs (resp. outputs) that could use this edge. Using Invariant 3.2, a total of $2^{l_i} \log N$ messages could use edge e_i. Therefore, the total number of ways of choosing q-tuples of messages that could pass through e_i is $\begin{pmatrix} 2^{l_i}k \\ q \end{pmatrix}$. Therefore, the total number of ways of choosing q-tuples of messages for all the edges $e_i, 1 \leq i \leq t$, is at most

$$\prod_i \begin{pmatrix} 2^{l_i}k \\ q \end{pmatrix} \leq k^{qt} \prod_i 2^{l_i q}$$

The probability that each chosen q-tuple of messages passes through the respective edge with the right color is

$$\frac{1}{\Delta^{qt}} \cdot \prod_i 2^{-(l_i+1)q}$$

where $\Delta = \beta k \log^{\frac{1}{q}} N$. Note that we can multiply probabilities, since the messages are distinct. Thus the probability that q messages of color c_i pass through each edge e_i is at most

$$k^{qt} \cdot \prod_i 2^{l_i q} \cdot \prod_i 2^{-(l_i+1)q} \cdot \frac{1}{\Delta^{qt}} \leq (\frac{k}{2\Delta})^{qt}$$

Since there are $(2 \log N)^t$ ways of choosing the edges $e_1, e_2, \cdots e_t$ on the paths $P_1, P_2, \cdots, P_t$ respectively, the net probability that all messages $m_1, \cdots, m_t$ fail to reach their destination is at most

$$(2 \log N)^t (\frac{k}{2\Delta})^{qt} \leq \frac{2^t}{(2\beta)^{qt}} \leq \frac{1}{2^{12t}}$$

for a sufficiently large value of β. □

Lemma 3.4 *Let each input (resp. output) have k messages in the beginning of a round. With high probability, there are at least $\frac{3}{4}k$ messages in each input (resp. output) with distinct colors.*

Proof: The probability that fewer than $\frac{3}{4}k$ colors are used at a specific input is at most

$$\begin{pmatrix} \Delta \\ \frac{3k}{4} \end{pmatrix} \cdot \left(\frac{(3k/4)}{\Delta}\right)^k \leq \left(\frac{\Delta e}{3k/4}\right)^{3k/4} \cdot \left(\frac{(3k/4)}{\Delta}\right)^k$$

where $\Delta = \beta k \log^{\frac{1}{q}} N$. Simplifying it further and using the fact that $k \geq \log N$, this probability is at most

$$\left(\frac{3e^3}{4\beta \log^{\frac{1}{q}} N}\right)^{k/4} \leq \frac{1}{N^c}$$

for some suitably large constant c. The probability that some input (resp. output) uses smaller than $\frac{3k}{4}$ colors is at most $N \cdot \frac{1}{N^c} = \frac{1}{N^{c-1}}$, which is a polynomially small probability. □

Lemma 3.5 *Assume that Invariant 3.2 holds in the beginning of the round. At the end of the round, the total number of messages in each input (resp. output) is at most $k/2$, with high probability.*

Proof: Let each input (resp. output) have k messages in the beginning of a round. Using Lemma 3.4, at least $\frac{3}{4}k$ of the messages in each input (resp. output) pick different colors, with high probability. We claim that, with high probability, at least $\frac{k}{2}$ of the messages that picked different colors succeed in reaching their destinations. To prove this claim, we use Lemma 3.3 to bound the probability that there exists a subset of $\frac{k}{4}$ messages with different colors that fail to reach their destination. This probability is at most

$$\left(\begin{array}{c} 3k/4 \\ k/4 \end{array} \right) \frac{1}{2^{12k/4}} \leq \frac{1}{2^{2k}}$$

which is polynomially-small, since $k \geq \log N$. □

Theorem 3.6 *If Invariant 3.2 holds in round i, it continues to hold in round $i+1$, with high probability.*

Proof: Using Lemma 3.5, we know that if the Invariant 3.2 holds at the beginning of round i, there are at most $k/2$ messages per input (resp. output) at the end of the round, with high probability. Since two copies are made of each remaining message, the total number of messages per input (resp. output) after Step 1 of round $i+1$ is at most k. Thus Invariant 3.2 holds in round $i+1$. □

We can now use these results to prove the main theorem.

Proof of Theorem 3.1: First, we show that at least one copy of each message succeeds in reaching its destination, with high probability. Let p be a message such that no copy of p got through till the end of the last-but-one round. In the last round, there will be $2^{2\log\log(Nk)} = \log^2(Nk)$ copies of p, of which at least $\log(Nk)$ copies will received different colors, with high probability (This follows from an argument similar to the proof of Lemma 3.4). The probability that none of the $\log(Nk)$ copies gets through in last round is at most

$$\frac{1}{2^{12\log(Nk)}} \leq \frac{1}{(Nk)^{12}}$$

Thus, the probability that there exists some message that did not reach its destination is at most

$$Nk \cdot \frac{1}{(Nk)^{12}} \leq \frac{1}{(Nk)^{11}}$$

i.e., this probability is polynomially small.

Next, we analyze the total time taken by the algorithm. It is possible to pipeline the sub-rounds of each round so that one sub-round starts every b steps. Note that the messages belonging to different sub-rounds of the same round never meet since any delayed message is removed from the network. Therefore, after $2\log N$ time, one subround completes every b steps. There are $\Delta = \beta k \log^{\frac{1}{q}} N$ sub-rounds in each round. Therefore, each round takes $bq\beta k \log^{\frac{1}{q}} N + 2q \log N$ bit-steps, which is $O(bqk \log^{\frac{1}{q}} N)$ bit-steps. Since there are $2\log\log(Nk)$ rounds, the total time taken is $O(bqk \log^{\frac{1}{q}} N \log\log(Nk))$ which is $O(bqk(\log^{\frac{1}{q}} N)w_1(n,k))$ bit-steps. □

3.2 Butterfly lower bound

In this section, we provide a lower bound on the time taken by any routing algorithm to route a random problem with k messages per input. The lower bound deals *only* with the class of algorithms that route the messages in *one pass* through the butterfly. Therefore, the lower bound does not apply to the algorithm presented in Section 3.1. The lower bound also extends to routing a random k-relation. In this section, and this section alone, we assume that q is a constant.

The proof of this lower bound generalizes the lower bound for $q = 1$ presented by Ranade, Schleimer, and Wilkerson[34]. The structure of our proof follows very closely the structure of theirs. Note that the lower bound matches our upper bound within small factors.

Theorem 3.7 *With high probability, any one-pass butterfly routing algorithm must take time T to route a random routing problem with k messages per input, where*

$$T = \Omega(bkl^{\frac{1}{q}}/w_2(N,k))$$

where $l = \min\{b, \log N\}$, and $w_2(N,k)$ is a slowly-growing function of N and k.

Throughout this proof we will consider only the truncated-butterfly that consists of the first l levels of the original $\log N$-level butterfly, where $l = \min\{b, \log N\}$. Any routing algorithm on the original butterfly translates in an obvious way to a routing algorithm on the truncated-butterfly that routes messages in the same or fewer time steps. We prove a lower bound of T for routing on the truncated-butterfly, which translates to a lower bound on the original butterfly.

Definition 3.8 *A set of B messages is said to collide if there exist $q+1$ messages in the set whose paths all use a single edge in the truncated-butterfly.*

The proof of Theorem 3.7 proceeds in two steps. First, we show that *every* set of $B = N\log^{2/q}(k\log N)/l^{\frac{1}{q+1}}$ messages chosen from the total of Nk messages collide, with high probability (Theorem 3.11). Next, in Theorem 3.12, we show that if the

routing algorithm completes in time T then there exists a set of B messages that do not collide – which, using Theorem 3.11 is a low probability occurrence.This implies the result of Theorem 3.7.

We need the following lemmas to prove Theorem 3.11.

Lemma 3.9 *Let m balls be thrown independently and randomly into n bins, where $m \leq n$. The probability that no bin receives more than q balls is at most*

$$\exp\left\{-\alpha m^{q+2}/n^{q+1}\right\}$$

where α is some constant that depends only on q.

Proof: Inspect each bin sequentially starting from bin 1. The probability P that no bin receives more than q balls is

$$P = P_1 \cdot P_2 \cdot P_n \tag{2}$$

where P_i is the probability that bin i receives at most q balls given that all bins from 1 to $i-1$ received at most q balls.

We examine bin i, after examining in order the first $i-1$ bins. If $i \leq \frac{m}{2q}$, at least $m/2$ balls are yet to be assigned to bins. Therefore, the probability that bin i receives more than q balls is at least

$$\sum_{i \geq q+1} \binom{m/2}{i} \frac{1}{n^i} \left(1 - \frac{1}{n}\right)^{m/2-i} \geq \binom{m/2}{q+1} \frac{1}{n^{q+1}} \left(1 - \frac{1}{n}\right)^{m/2} \geq \alpha' \frac{m^{q+1}}{n^{q+1}} \tag{3}$$

where α' is a constant that depends only on q. Thus, for all $i \leq m/2q$,

$$P_i \leq 1 - \alpha' \frac{m^{q+1}}{n^{q+1}} \tag{4}$$

We use the trivial upper bound of 1 for all P_i, $i > m/2q$. Now, using equations 2 and 4, we obtain

$$P \leq \left(1 - \alpha' \frac{m^{q+1}}{n^{q+1}}\right)^{m/2q} \leq \exp\left\{-\alpha m^{q+2}/n^{q+1}\right\}$$

where α is a constant depending only on q.

Lemma 3.10 *Given an arbitrary set of $B = N \log^{2/q}(k \log N)/l^{\frac{1}{q+1}}$ messages, the probability that these messages do not collide is at most*

$$\exp\left\{-\alpha \frac{B^{q+2} b}{N^{q+1} \log\log N}\right\}$$

Proof: We partition the truncated sub-butterfly into $b/\log n$ strips, where $n = \log N$. The j^{th} strip consists of all nodes at levels $j \log n$ through to $(j+1)\log n$.

First, we bound the probability that the B messages do not collide in a specific strip S. Strip S consists of N/n n-input sub-butterflies. Let B_i, $1 \leq i \leq N/n$, represent the number of messages that originate in the inputs of the i^{th} sub-butterfly in strip S. Each message routes to a random output of the sub-butterfly. Therefore the probability that these B_i messages fail to collide at the outputs of the sub-butterfly can be upper bounded, using Lemma 3.9, to be at most

$$\exp\left\{-\alpha B_i^{q+2}/n^{q+1}\right\}$$

Since the sub-butterflies within strip S are disjoint, the events of collision within each sub-butterfly are independent. Therefore, the probability that there is no collision in strip S is at most

$$\prod_{i=1}^{i=N/n} \exp\left\{-\alpha B_i^{q+2}/n^{q+1}\right\}.$$

The above expression is maximized when all the B_i are equal to BN/n. Substituting this value for the B_i's, the probability of that there is no collision in strip S is at most

$$\exp\left\{-\alpha((Bn/N)^{q+2}/n^{q+1})\frac{N}{n}\right\} = \exp\left\{-\alpha B^{q+2}/N^{q+1}\right\}.$$

The probability of a collision at an output of some sub-butterfly in a strip S depends on the distribution of messages to the sub-butterflies within S. The lemma would be easier to prove if the probability of a collision within S was independent of the probability of a collision within any other strip. This is not true. However, because we count collisions only at the outputs of the sub-butterflies, within a sub-butterfly it does not matter which inputs start with messages. Furthermore, the probability of a collision within S is maximized when BN/n messages start at the inputs of each sub-butterfly in S. Hence, the bound that we derived for this case holds independent of the conditioning of the distribution of messages at the inputs of the sub-butterflies in S. As a consequence, when using this worst-case bound, we can assume that the probability of a collision within one strip is independent of the probability of a collision with any other strip. There are a total of $b/\log n$ strips. Therefore, the probability that there is no collision in the entire truncated butterfly is at most

$$\exp\left\{-\alpha\frac{B^{q+2}b}{N^{q+1}\log\log N}\right\}.$$

□

Theorem 3.11 *Every set of B messages chosen from the total of Nk messages collides, with high probability, where $B = N\log^{2/q}(k\log N)/l^{\frac{1}{q+1}}$.*

Proof: There a total of $\begin{pmatrix} Nk \\ B \end{pmatrix}$ choices for the B messages. Thus, using Lemma 3.10, the probability that there exists a set of B messages that do not collide is at most

$$\begin{aligned}
& \exp\left\{-\alpha\frac{B^{q+2}b}{N^{q+1}\log\log N}\right\}\begin{pmatrix} Nk \\ B \end{pmatrix} \\
\leq\ & \exp\left\{-\alpha\frac{B^{q+2}b}{N^{q+1}\log\log N}\right\}\left(\frac{Nke}{B}\right)^{B} \\
=\ & \exp\left\{-B(\alpha\frac{B^{q+1}b}{N^{q+1}\log\log N}-\log(Nke/B))\right\} \\
\leq\ & \exp\left\{-B(\alpha\log^{1+2/q}(k\log N)-O(\log(k\log N)))\right\} \\
\leq\ & \exp\{-\omega(B)\} \leq \frac{1}{N^{\omega(1)}}
\end{aligned}$$

(Recall that $B = N\log^{2/q}(k\log N)/l^{\frac{1}{q+1}}$ and $l = \min\{b, \log N\}$.) Thus, the probability that there exists a set of B messages that do not collide is polynomially small. □

Theorem 3.12 *If an algorithm takes T time to route a total of Nk messages then there exists a set of Nkb/T messages that do not collide.*

Proof: This lemma and its proof is similar to Lemma 4 of [34]. The key is that the messages can be partitioned into phases such that the heads of all messages in phase i arrive at the last level of the truncated-butterfly at time $l + ib$. Without loss of generality, assume that every message that arrives after time l is delayed by some other message – any message p that arrives after time l without incurring delay can be started off at an earlier time until either p is delayed by some other message or p arrives at time l.

Sort all the messages in the non-decreasing order of their time of arrival. Now, let p be the first message in the list that arrives out of phase, i.e., it arrives at a time other than $l + ib$ for some integer i. Let p' be the last message that delayed message p. It is easy to see that as soon as the tail of p' moves out of the path of p, the head of p starts moving and is always one level behind the tail of p'. This implies that the head of p arrives b steps after the head of p'. Since p' arrives earlier than p, by assumption, p' arrives in phase. This implies that p arrives in phase also, which is a contradiction.

The fact that we can partition all the messages into T/b phases imply that at least one phase has Nkb/T messages. □

We can now prove the main lower bound theorem.

Proof of Theorem 3.7: From Theorem 3.11, we know that the probability that

there exists a set of B messages that do not collide is small. Therefore, setting $B = Nkb/T$, and using Theorem 3.12, the routing algorithm must take time

$$T = Nkb/B = Nkb \cdot (l^{\frac{1}{q+1}}/N \log^{2/q}(k \log N)).$$

Simplifying further, we get

$$T \geq bkl^{\frac{1}{q}}/w_2(N,k),$$

where $w_2(N,k) = O(l^{1/q^2} \log^{2/q}(k \log N))$. □

4 Acknowledgements

The authors would like to thank Manos Varvarigos, who suggested examining the effect of queue size on the worst-case performance of wormhole routing algorithms.

References

[1] W. A. Aiello, F. T. Leighton, B. M. Maggs, and M. Newman. Fast algorithms for bit-serial routing on a hypercube. *Mathematical Systems Theory*, 24(4):253–271, 1991.

[2] N. Alon. A parallel algorithmic version of the Local Lemma. *Random Structures and Algorithms*, 2(4):367–378, 1991.

[3] S. Arora, T. Leighton, and B. Maggs. On-line algorithms for path selection in a non-blocking network. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 149–158, May 1990.

[4] Butterfly^{TM} Parallel Processor Overview. BBN Report No. 6148, Version 1, BBN Advanced Computers, Inc., Cambridge, MA, March 1986.

[5] J. Beck. An algorithmic approach to the Lovász Local Lemma I. *Random Structures and Algorithms*, 2(4):343–365, 1991.

[6] B. Beizer. The analysis and synthesis of signal switching networks. In *Proceedings of the Symposium on Mathematical Theory of Automata*, pages 563–576, Brooklyn, NY, 1962. Brooklyn Polytechnic Institute.

[7] V. E. Beneš. Optimal rearrangeable multistage connecting networks. *Bell System Technical Journal*, 43:1641–1656, July 1964.

[8] S. Borkar, R. Cohn, G. Cox, S. Gleason, T. Gross, H. T. Kung, M. Lam, B. Moore, C. Peterson, J. Pieper, L. Rankin, P.S. Tseng, J. Sutton, J. Urbanski, and J Webb. iWarp, an integrated solution to high-speed parallel computing. In *Proceedings of the 1988 International Conference on Supercomputing*, pages 330–339, November 1988.

[9] A. Borodin and J. E. Hopcroft. Routing, merging, and sorting on parallel models of computation. *Journal of Computer and System Sciences*, 30(1):130–145, February 1985.

[10] A. Borodin, P. Raghavan, B. Schieber, and E. Upfal. How much can harware help routing? In *Proceedings of the 25th Annual ACM Symposium on the Theory of Computing*, pages 573–591, May 1993.

[11] R. Cypher. Minimal, deadlock-free routing in hypercubic and arbitrary networks. In *Proceedings of the 7th IEEE Symposium on Parallel and Distributed Processing*, pages 122–129, October 1995.

[12] R. Cypher and L. Gravano. Storage-efficient, deadlock-free, adaptive packet routing algorithms for torus networks. *IEEE Transactions on Computers*, 43(12):1376–1385, 1994.

[13] W. Dally and C. Seitz. Deadlock free message routing in multiprocessor interconnection networks. *IEEE Transactions on Computers*, C–36(5):547–553, May 1987.

[14] W. J. Dally. Performance analysis of k-ary n-cube interconnection networks. *IEEE Transactions on Computers*, 39(6):775–785, 1990.

[15] W. J. Dally. Virtual-channel flow control. *IEEE Transactions on Parallel and Distributed Systems*, 3(2):194–205, March 1992.

[16] W. J. Dally and C. L. Seitz. The Torus Routing Chip. *Distributed Computing*, 1(4):187–196, 1986.

[17] S. Felperin, P. Raghavan, and E. Upfal. A theory of wormhole routing in parallel computers. In *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science*, pages 563–572, October 1992.

[18] C. Kaklamanis, D. Krizanc, and A. Tsantilas. Tight bounds for oblivous routing in the hypercube. In *Proceedings of the 2nd Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 31–36, July 1990.

[19] R. R. Koch. Increasing the size of a network by a constant factor can increase performance by more than a constant factor. In *Proceedings of the 29th Annual Symposium on Foundations of Computer Science*, pages 221–230. IEEE Computer Society Press, October 1988.

[20] R. K. Koeninger, M. Furtney, and M. Walker. A shared MPP from Cray research. *Digital Technical Journal*, 6(2):8–21, Spring 1994.

[21] C. P. Kruskal and M. Snir. The performance of multistage interconnection networks for multiprocessors. *IEEE Transactions on Computers*, C–32(12):1091–1098, December 1983.

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

[23] 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(1):157–205, July 1994.

[24] F. T. Leighton, B. M. Maggs, and S. B. Rao. Packet routing and job-shop scheduling in O(congestion + dilation) steps. *Combinatorica*, 14(2):167–180, 1994.

[25] T. Leighton. Methods for message routing in parallel machines. In *Proceedings of the 24th Annual ACM Symposium on the Theory of Computing*, pages 77–96, May 1992.

[26] T. Leighton and B. Maggs. Fast algorithms for finding O(congestion + dilation) packet routing schedules. In *Proceedings of the 28th Hawaii International Conference on System Sciences*, January 1995. to appear.

[27] C. E. Leiserson, Z. S. Abuhamdeh, D. C. Douglas, C. R. Feynman, M. N. Ganmukhi, J. V. Jill, W. D. Hillis, B. C. Kuszmaul, M. A. St. Pierre, D. S. Wells, M. C. Wong, S.-W. Yang, and R. Zak. The network architecture of the connection machine CM-5. In *Proceedings of the 4th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 272–285, June 1992.

[28] B. M. Maggs and R. K. Sitaraman. Simple algorithms for routing on butterfly networks with bounded queues. In *Proceedings of the 24th Annual ACM Symposium on the Theory of Computing*, pages 150–161, May 1992.

[29] Y. Mansour and B. Patt-Shamir. Greedy packet scheduling on shortest paths. *Journal of Algorithms*, 14:449–65, 1993.

[30] P. M. Merlin and P. J. Schweitzer. Deadlock avoidance in store-and-forward networks. 1: Store-and-forward deadlock. *IEEE Transactions on Communications*, 28:345–354, 1980.

[31] F. Meyer auf der Heide and B. Vöcking. A packet routing protocol for arbitrary networks. In *Proceedings of the 12th Symposium on Theoretical Aspects of Computer Science*, March 1995. To appear.

[32] M. D. Noakes, D. A. Wallach, and W. J. Dally. The J-Machine multicomputer: an architectural evaluation. In *Proceedings of the 20th Annual International Symposium on Computer Architecture*, pages 224–235, May 1993.

[33] G. D. Pifarré, L. Gravano, S. A. Felperin, and J. L. C. Sanz. Fully-adaptive minimal deadlock-free packet routing in hypercubes, meshes, and other networks.

In *Proceedings of the 3rd Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 278–290, July 1991.

[34] A. Ranade, S. Schleimer, and D. S. Wilkerson. Nearly tight bounds for wormhole routing. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, 1994.

[35] C. L. Seitz, W. C. Athas, C. M. Flaig, A. J. Martin, J. Seizovic, C. S. Steele, and W.-K. Su. The architecture and programming of the ametek series 2010 multicomputer. In *Proceedings of the 3rd Conference on Hypercube Concurrent Computers and Applications*, volume 1, pages 33–36, 1988.

[36] J. Spencer. *Ten Lectures on the Probabilistic Method*. SIAM, Philadelphia, PA, 1987.

[37] L. G. Valiant. A scheme for fast parallel communication. *SIAM Journal on Computing*, 11(2):350–361, May 1982.

[38] A. Waksman. A permutation network. *Journal of the ACM*, 15(1):159–163, January 1968.