On the Benefit of Supporting Virtual Channels in Wormhole Routers

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

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

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

Abstract

This paper analyzes the impact of virtual channels on the performance of wormhole routing algorithms. We show that in any network in which each physical channel, i.e., communication link, can support up to B virtual channels, it is possible to route any set of messages with L flits each, whose paths have congestion C and dilation D in $(L + D)C(D \log D)^{1/B} 2^{O(\log^*(C/D))}/B$ flit steps, where a flit step is the time taken to transmit a single flit across a link. We also prove a nearly matching lower bound, i.e., for any values of C, D, B, and L, where $C, D \geq B + 1$ and $L = (1 + \Omega(1))D$, we show how to construct a network and a set of L-flit messages whose paths have congestion C and dilation D that require $\Omega(LCD^{1/B}/B)$ flit steps to route. These upper and lower bounds imply that increasing the buffering capacity and the bandwidth of each physical channel by a factor of B can speed up a wormhole routing algorithm by a superlinear factor, i.e., a factor significantly larger than B. We also present a simple randomized wormhole routing algorithm for the butterfly network. The

[*]Richard Cole is supported in part by NSF Grants No. CCR-92-02900 and CCR-95-03309.

[†]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.

[‡]Ramesh Sitaraman is supported in part by NSF Grant No. CCR-94-10077 and by an NSF CAREER Award No. CCR-97-03017.

algorithm routes any q-relation on the inputs and outputs of an n-input butterfly in $O(L(q+\log n)\log^{1/B} n \log\log(qn)/B)$ flit steps. We present a nearly-matching lower bound that holds for a broad class of algorithms.

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 [36], and commercial machines such as the Intel Paragon, Cray T3D [23], and Connection Machine CM-5 [31].

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 buffered at a node of the network. Typically the number of bits in a flit is small. The message is viewed as a worm that traverses the network in a flit-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 buffer flits is limited to one flit 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. Furthermore, since a switch can only buffer one flit of a message, if the header flit of a message cannot advance then the flits following the header must stall.

A natural generalization of wormhole routing is to allow a switch to buffer more than one flit per message, perhaps even allowing it to buffer an entire message. This approach is called *virtual cut-through* routing [21], and predates wormhole routing.

A third common routing method is *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 from switch to switch as it traverses its path. The time taken to transmit an entire message across a single link is called a *message step*.

Wormhole and virtual-cut through routing have several advantages over store-and-forward routing. In a store-and-forward router, if a message traverses a path of length D, and is never delayed, then it will reach its destination in D message steps. If the message contains L flits, each message step equals L flit steps, where a *flit step* is defined to be the time taken to transmit a single flit across a single link. Thus, the total time for a message to reach its destination in store-and-forward routing is LD flit steps. In a wormhole (or virtual cut-through) router, however, the header flit does not wait for the rest of the message. Assuming that the header flit is never delayed, it arrives at its destination after D flit steps, and the last flit of the message arrives after $D + L - 1$ flit 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 message 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 [14], 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 buffer 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 buffer 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 buffer 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, and each physical channel of the J-Machine supports 2 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 emulate B virtual channels can yield a speed up that is larger than linear in B. These analytical results suggest to the practical network designer that adding even a small number of virtual channels to a wormhole network can disproportionately enhance the routing performance of the network.

1.1 Congestion, dilation, message length, and virtual channels

Throughout this paper we express the performance of routing algorithms in terms of four parameters: congestion, dilation, message length, and buffer size. We measure routing time in flit steps, where a flit step is the time needed to transmit a single flit across a single link (or, edge).

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* L is simply the number of flits in each message, including the header flit that contains information required to route the message through the network.

Throughout this paper, we assume that each physical channel (or, edge) of the network supports B *virtual channels*. A flit that is transmitted across an edge of the network is stored in a buffer at the head of that edge. Each buffer can hold a maximum of B flits, each flit belonging to a different message. Further, each physical channel can transmit B flits, one from each virtual channel, *simultaneously*. The header flit of a message cannot be transmitted across an edge if there is no buffer space available at the head of that edge. In this case, the flits of the message are stalled and the message is said to be *blocked*.

Each message is initially stored in an *injection buffer* that is external to the network at its source node. The message is injected by the source node one flit at a time into the network. As soon as a flit reaches its destination node, the flit is removed from the network and stored in a *delivery buffer* that is also external to the network. The injection buffers and the delivery buffers are assumed to be sufficiently large to store messages in their entirety.

1.2 The butterfly network

Some of the results in this paper 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 ($0 \leq i \leq \log n$) 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 $\log n$.) 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 *q-relation routing problem* and the *random routing problem* with q messages per input. In a q-relation routing problem at most q messages originate at each input and at most q messages are destined for each output. The special case of $q = 1$ is called *permutation* routing. In a random routing problem with q messages per input, each of the q messages at each of the inputs independently selects 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 [28], or his book [25]. In what follows we describe only the research most closely related to this paper.

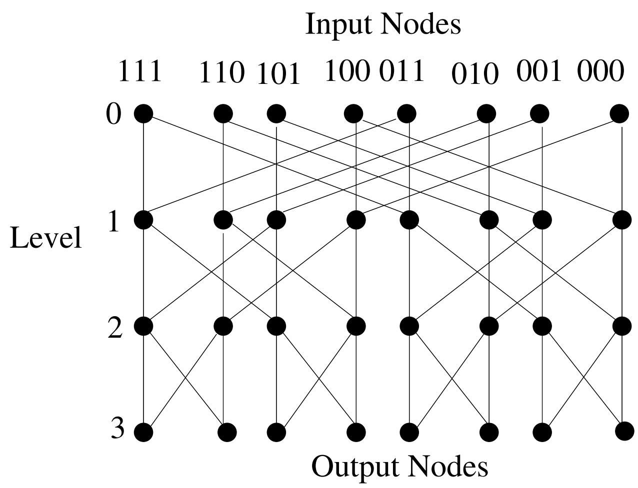

Figure 1: An 8-input butterfly network.

1.3.1 Network-independent algorithms

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 [27]. 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 buffer at the head of an edge. The $O(C+D)$ bound improves on the naive $O(CD)$ bound, and is optimal. Since a message step equals L flit steps, the bound translates to $O(L(C + D))$ flit steps, which is optimal for flit-serial routing when $C > D$ (because some edge must transmit LC flits). 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, i.e., the proof shows the existence of a schedule without providing an efficient algorithm for actually computing the schedule. Leighton et al. also devised a simple randomized on-line store-and-forward algorithm that routes n messages in $O(C + D \log n)$ message steps, with high probability. Scheideler recently presented in [41] an alternative simpler proof of the existence of $O(C + D)$-step schedules that only require edge buffers of size 2.

The $O(C + D)$ bound of Leighton et al. was followed by a number of algorithmic results. For the special case of leveled networks, Leighton, Maggs, Ranade, and Rao [26] presented a simple on-line store-and-forward algorithm for routing any set of n messages in a leveled network with depth D in $O(C + D + \log n)$ message steps. In a *leveled* network with depth D, each node is labeled with an integer between 0 and D, and each edge with its tail on level i, $0 \leq i < D$, has its head on level $i + 1$. Mansour and Patt-Shamir [33] then showed that, in any network, if messages are routed greedily on shortest paths, then all of the messages reach their destinations within $D + n - 1$ message 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 [35] later devised a simple on-line randomized algorithm that routes all messages to their destinations in $O(C+D+\log n)$ message steps, with high probability, provided that the paths taken by the messages are short-cut free (e.g., shortest paths). Recently, Leighton, Maggs, and Richa [29, 30] discovered a sequential 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 $O(P \log^{1+\epsilon} P \log^*(C+D))$ time, for any fixed $\epsilon > 0$, where P is the sum of the lengths of the paths taken by the messages.

Recent advances in on-line local control algorithms for universal store-and-forward routing include the work of Rabani and Tardos [39] and Ostrovsky and Rabani [37]. Ostrovsky and Rabani improve on the results in [39] by presenting a randomized on-line algorithm that delivers all the messages to their destinations in $O(C + D + \log^{1+\epsilon} N)$ message steps with high probability, for arbitrary $\epsilon > 0$.

The universal store-and-forward message routing results outlined so far deal only with the problem of scheduling messages after paths have been chosen for each message. Srinivasan and Teo [45] show how to select paths, given the source and destination for each message, so that value of $C + D$ is smallest possible to within constant factors. (Finding the exact minumum value of $C + D$ is NP-hard.) This result in conjunction with the results in [29, 30] yields the first offline algorithm for routing messages, given only the sources and the destinations of the messages, in time that is a constant factor from optimal using constant-sized buffers.

In contrast with store-and-forward routing, there is relatively little prior work on network independent wormhole routing algorithms. Greenberg and Oh [19] were the first to state non-trivial network-independent wormhole routing results in terms of L, C, and D. They created a randomized algorithm that takes $O(lCD + lCL\log n)$ flit steps, where $l = \min\{L, D\}$, provided the paths of any two messages intersect in at most one contiguous sequence of edges and the channel dependency graph is acyclic to avoid deadlocks. Ranade, Schleimer, and Wilkerson [40] then showed that on any leveled network, any set of L-flit messages whose paths have congestion C and dilation D can be routed in $O(LCD)$ flit steps. The $O(LCD)$ bound improves on the naive $O((L+D)CD)$ bound[1]. Neither of these papers considered the case $B > 1$.

Cypher, Meyer auf der Heide, Scheideler, and Vöcking [13] have recently independently proved a number of results that are closely related, and in some cases superior, to the results in this paper. They give a simple randomized algorithm for routing n messages in $O((LCD^{1/B}+(L+D)\log n)/B)$ flit steps, for any $B \leq \log n$. It is worth noting that their algorithm is on-line, in the sense that it can easily be performed by the network switches themselves, whereas the network-independent algorithm described in this paper is not on-line. While the bounds on the running time obtained by Cypher et al is superior to our bounds for some ranges of the parameters L, C, D, B, and n, our bounds are superior to theirs for other ranges of the parameters. The techniques used in the two papers to design network-independent algorithms are very different and are of independent interest.

Scheideler and Vöcking [42] have also recently shown that the same factor of $D^{1/B}$ appears in the maximum injection rate for continuous wormhole routing algorithms. A *continuous* routing algorithm is one that accepts packets that are randomly generated over time according to a Poisson process. The algorithms described in this paper, which are designed to route a batch of packets, are not continuous.

[1]The naive bound is obtained via a simple coloring argument. Suppose that we construct a graph with a node for each worm, and an edge between any two worms whose paths share an edge. The degree of this graph is at most $D(C-1)$, since each worm uses D edges, and might share each edge with as many as C-1 other worms. The graph can be colored with $D(C-1)+1$ colors, so that no two neighbors have the same color. These colors can then be used to derive a schedule. First route all worms with color 1, then all worms with color 2, and so on. For any color, no two worms intersect, so there is no waiting or chance of deadlock. Hence, any color can be routed in $L+D-1$ flit steps. This gives an overall time bound of O((L+D)(CD)) flit steps.

1.3.2 Network-independent lower bounds

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 algorithm 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 [20]. These bounds can be translated to lower bounds (in flit steps) for oblivious wormhole routing algorithms simply by multiplying the time by the message length (in flits), e.g., the Kaklamanis et al. bounds becomes $\Omega(L\sqrt{n}/d)$ flit 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]. The result of [1] adapted to our model implies that on any degree-d network, almost all permutations require $\Omega(\log n/(\log d+\log\log n))$ message steps, or $\Omega(L\log n/(\log d+\log\log n))$ flit steps. For constant-degree networks such as the butterfly, this bound is $\Omega(L\log n/\log\log n)$ flit 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. The bound from [10] is stronger than the bound from [1] when d is large.

For wormhole routing, Ranade, Schleimer, and Wilkerson [40] showed how to construct a network with queue size $B = 1$, and a set of L-flit messages whose paths have congestion C and dilation D in which the optimal wormhole routing schedule requires $\Omega(LCD)$ flit steps. Hence, in this example store-and-forward routing, which requires at most $O(L(C+D))$ flit steps, is more efficient than wormhole routing.

As a point of contrast with our paper, Cypher, Meyer auf der Heide, Scheideler, and Vöcking [13] do not present a network-independent lower bound.

1.3.3 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 [25, 28]. 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 [47] gave an elegant linear time 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 L-flit messages in $O(L+\log n)$ flit steps, or any q-relation in $O(qL+\log n)$ flit 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.

The next few results deal with the problem of circuit-switching on a butterfly network. In a circuit-switched network, each message must lock-down a dedicated path from its input node to its output node before it can be transmitted.

Kruskal and Snir [24] showed that if each input in an n-input circuit-switched 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 (i.e., succeed in locking-down paths) is $\Theta(n/\log n)$.

Koch [22] generalized the result of Kruskal and Snir by showing that if each edge can support B messages, then the expected fraction of messages that get through is $\Theta(n/\log^{1/B} n)$. Thus, Koch observed in the context of circuit-switching that increasing the capacity of each edge by a constant factor may increase the performance of a routing algorithm by more than a constant factor. Koch's work was motivated by the fact that on the BBN Butterfly parallel computer [4], $B = 2$. Note that in this paper we show similar super-linear resource-performance tradeoffs for wormhole routing.

Maggs and Sitaraman [32] generalized the previous two results by showing that, by making two passes through a butterfly, it is possible to route a $\Theta(n/\log^{1/B} n)$ fraction of any permutation (rather than only a random permutation), with high probability.

Some results derived in the context of circuit-switching can be readily adapted to wormhole routing. In Problem 3.285 of [25], Leighton describes an algorithm for solving a random routing problem in which each input has one L-flit message to send. The algorithm runs in $O((L + \log n) \log n)$ flit steps. In Problem 3.286, he observes that the algorithm can be converted to one that routes any permutation using Valiant's idea [46] of first routing to random intermediate destinations. For the interesting case of $L = O(\log n)$, the time is $O(\log^2 n)$ flit steps. The algorithm can easily be generalized to the case $q > 1$, and runs in $O(q(L+\log n)\log n)$ flit steps. For the interesting case of $L = O(\log n)$ and $q = \log n$, the time is $O(\log^3 n)$ flit steps.

Felperin et al. [18] independently discovered an $O(\log^4 n)$ flit step algorithm for solving a random problem for the case $L = O(\log n)$ and $q = \log n$, and then Ranade et al. [40] discovered an $O(\log^3 n \log\log n)$ flit step algorithm.

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

Cypher, et al. [13] have recently independently described a randomized algorithm that requires $O((qL\log^{1/B} n+\log(qn)\log n)/B)$ flit steps, with high probability, to route q worms from each input to random outputs, assuming that each edge can simultaneously transmit flits from B messages. They also proved an $\Omega((qL\log^{1/B} n(\log\log n)^{-2/B})/B)$ flit step lower bound. Both these bounds are similar to the bounds we prove in this paper.

1.3.4 Other wormhole routing algorithms

Following the paper by Dally and Seitz, many papers were written on the subject of wormhole routing. Most of these 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 [43], and Dally and Seitz [17] describe the Torus Routing Chip.

In the second category, two of the most influential papers were written by Dally [15, 16]. 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 [34], meshes [38], toruses [12], and hypercubes [38].

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 L-flit messages on an n-node hypercube in any permutation in $O(L + \log n)$ flit steps. The algorithm requires B 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 L-flit messages from the inputs to the outputs of an n-input network in $O(L+\log n)$ flit 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. [40] presented off-line algorithms for routing in $O(LC + D)$ flit steps. The lengths of the schedules produced by these algorithms are optimal.

1.4 Our results

Our most general result is a proof that if each edge can queue up to B message flits, then it is possible to route any set of L-flit messages whose paths are edge simple and have congestion C and dilation D in

$$(L + D)C(D \log D)^{1/B} 2^{O(\log^*(C/D))}/B$$

flit steps. The proof is nonconstructive, but can be made constructive using the techniques in [29, 30]. 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, B, and L, where $C, D \geq B+1$ and $L = (1+\Omega(1))D$, we show how to construct a network and a set of paths with congestion C and dilation D and a set of L-flit messages that require $\Omega(LCD^{1/B}/B)$ flit steps to route. The proof of our lower bound is a generalization of the $\Omega(LCD)$ lower bound proved by Ranade et al. for the case $B = 1$. These nearly 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 [22] for circuit-switching on the butterfly.

For the butterfly, we present a simple randomized wormhole routing algorithm. The algorithm routes a q-relation on the inputs and outputs of an n-input butterfly in $O(L(q+\log n)(\log^{1/B} n) \log\log(qn)/B)$ flit steps, when $B \leq \log\log n/\log\log\log n$. The algorithm has the same performance (to within small factors) as the fastest previously known algorithms when $B = 1$, but is faster by a factor of about $B \log^{1-1/B} n$ for $1 < B \leq \log\log n/\log\log\log n$. 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 $L = \log n$, $q = \log N$, and $B = 1$, the lower bound of Ranade et al. is $\Omega(\log^3 n/(\log\log n)^2)$ flit steps, but several known store-and-forward routing algorithms run in $O(\log^2 n)$ flit steps (see Leighton[25]). Hence, the store-and-forward algorithms are faster. The switches executing the store-and-forward algorithms, however, must each have the capacity to buffer an entire message, i.e., $\Omega(\log n)$ flits. For $B = \log\log n/\log\log\log n$, the time for our wormhole routing algorithm drops to $O(\log^2 n \log\log n \log\log\log n)$. Hence, a wormhole routing algorithm can achieve nearly the same time bound as the store-and-forward algorithm, while buffering at most $\log\log n/\log\log\log n$ flits at the end of each edge.

For a fixed amount of buffer space, it is also interesting to compare the performance of a wormhole router with virtual channels to a virtual cut-through router. Suppose that for each incoming edge, a switch in the wormhole router can store up to one flit from each of B different messages, and that in the virtual cut-through router, for each incoming edge, a switch can store up to B flits from a single message. The performance of the virtual cut-through router will be roughly equivalent to the performance of a wormhole router in which there are no virtual channels, but in which the messages have length L/B, rather than L. Hence the speedup is linear in B. In the

wormhole router with virtual channels, however, the speedup is $BD^{1-1/B}$, which can be much larger when B is small and D is large.
Remarks. The algorithms presented in this papers also hold for a slightly different model of virtual channels where each switch can buffer B flits but each switch can forward only one flit per time step (as opposed to B flits per time step). In this model, the buffering is increased by a factor of B (in comparison with a network with no virtual channels), but the bandwidth of the links remain the same. The two algorithms presented in this paper can be emulated in this more restrictive model with a slowdown of a factor of B. Interpreting the results in this restricted model, our work implies that increasing the size of the buffers alone by a factor of B decreases the running time bound of our network-independent (resp., butterfly) routing algorithm by a factor of about $D^{1-1/B}$ (resp., $\log^{1-1/B} N$). Note that this factor could be larger than B yielding a superlinear benefit.

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 provide offline algorithms for wormhole routing and bound their routing times in terms of the parameters L, C, D, and B.

In Section 2.2 we show that there exists a routing problem that requires at least $\Omega(LCD^{1/B}/B)$ bit steps to route. The lower bound closely matches the upper bounds derived in Section 2.1.

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 "sufficient" length so that messages with different colors do not "interfere" with each other. Our goal is to produce a coloring so that at most B messages of the same color use any given edge. This provides a schedule where the messages make progress without getting blocked, since B messages can be multiplexed over any given edge at any given time step.

The coloring is obtained by repeated refinement. Each step of the construction uses the Lovász Local Lemma (LLL) [44, pp. 57–58] outlined below.

Lemma 2.1.1 (Lovász) *Let $A_1, \cdots, A_r$ be a set of "bad" events each occurring with probability at most p. Suppose that every bad event depends on at most b other bad events (i.e., every bad event is mutually independent of some set of $r - b$ other bad events). If $4pb < 1$, then the probability that no bad event occurs is nonzero.*

The following definitions will be helpful.

Definition 2.1.2 *A color class consists of all messages of a given color.*

Definition 2.1.3 *Given a set S of messages with a color assigned to each message in S, the* multiplex size *of S is the maximum over all edges and all color classes of the number of messages in a color class crossing an edge.*

Initially, all messages are placed in a single color class. So the messages have multiplex size C. Our goal is to reduce the mutliplex size from C to B by iteratively applying the following lemma.

Lemma 2.1.4 (Color Refinement) *Given a collection of messages with multiplex size at most m_s, it is possible to partition each original color class into r new color classes so that the multiplex size is at most m_f, provided that one of the following conditions hold.*

1. $\log D \geq m_s > B$, $m_f = B$, *and* $r = 8(D \cdot m_s)^{1/B} m_s/B$.

2. $D \geq m_s > \log D$, $m_f = \log D$, *and* $r = 32 m_s / \log D$.

3. $m_s = x > D$, $m_f = \max\{D, \log x\}$, *and* $r = 32x / \max\{D, \log x\}$.

Proof: Each original color class is partitioned into r new color classes in the following fashion. (Note that the r new colors used to refine an original color class are distinct from the new colors used to refine other original color classes.) Each message in the original color class is equally likely to be placed in any of the r new color classes and the choice for each message is independent of the choices made for any other message.

We show that there exists a coloring with multiplex size m_f by using Lemma 2.1.1. For each new color class and edge of the network, we say that a bad event has occurred if more than m_f messages of that color class use that edge. The probability of a bad event is clearly at most $p = \binom{m_s}{m_f}/r^{m_f}$. The occurrence of a bad event gives information about the colors of at most m_s messages. Since each of these messages use at most D edges, each bad event influences at most $b = m_s \cdot D$ other bad events. Now, since in each of the three cases in the lemma,

$$4pb = 4\binom{m_s}{m_f} m_s D/r^{m_f} < 1,$$

we can apply Lemma 2.1.1 to show that the probability that no bad event occurs is non-zero. The lemma follows. □

Theorem 2.1.5 *Given a routing problem with parameters L,C,D, and B, there is a wormhole routing schedule of length*

1. $O((L+D)C(D \cdot C)^{1/B}/B)$ *flit steps, for* $C < \log D$,
2. $O((L+D)C(D\log D)^{1/B}/B)$ *flit steps, for* $\log D \leq C \leq D$, *and*
3. $(L+D)C(D\log D)^{1/B}2^{O(\log^*(C/D))}/B$, *for* $C > D$.

Proof: Initially, all messages are placed in a single color class. So the messages have multiplex size C. Then, we reduce the mutliplex size from C to B by iteratively applying Lemma 2.1.4. Finally, we route each color class such that messages in different color classes do not interfere with each other. Since at most B messages of a given color use any edge, all the messages in a given color class can be routed simultaneously without any message being blocked. A message that is never blocked takes at most $(L+D-1)$ flit steps to complete routing, since the header flit takes at most D flit steps to reach its destination and $L-1$ more flit steps are required for the other flits to reach the destination. Thus, we start routing the messages in the i^{th} color class at time $(i-1)(L+D-1)$ and we can complete routing all the messages in time $\kappa(L+D-1)$, where κ is the total number of color classes.

We compute the number of color classes κ and the resultant schedule length for each of the following cases.

Case 1: $C < \log D$**.** In this case, we set $m_s = C$, $m_f = B$ and $r = 8(D \cdot C)^{1/B}C/B$ and apply condition 1 of Lemma 2.1.4. The length of the schedule that we obtain is

$$(L + D - 1) \cdot 8(D \cdot C)^{1/B}C/B = O((L + D)C(D \cdot C)^{1/B}/B).$$

Case 2: $\log D \le C \le D$**.** In this case, a two-step construction is used. In the first step, the multiplex size of the messages is reduced from C to $\log D$ using condition 2 of Lemma 2.1.4. Note that we replace each color class with $r = 32C/\log D$ color classes in this step. In the second step, the multiplex size is reduced from $\log D$ to B using condition 1 of Lemma 2.1.4. Note that each color class is replaced with $r = 8 \log D(D \log D)^{1/B}/B$ color classes. Overall, the number of color classes used is

$$(32C/\log D) \cdot (8 \log D(D \log D)^{1/B}/B) = O(C(D \log D)^{1/B}/B).$$

This gives a schedule of length $O((L + D)C(D \log D)^{1/B}/B)$.
Case 3: $C > D$**.** In this case a three-step construction is used. In the first step, the multiplex size of the messages is reduced from C to D by iteratively applying condition 3 of Lemma 2.1.4, $O(\log^* C - \log^* D)$ times. The number of color classes created in this step is at most

$$32^{O(\log^* C - \log^* D)}C/D = 2^{O(\log^*(C/D))}C/D.$$

In the second step, the multiplex size is reduced from D to $\log D$ using condition 2 of Lemma 2.1.4. In this step, each color class is replaced with $32D/\log D$ color classes. In the third step, the multiplex size is reduced from $\log D$ to B using condition 1 of Lemma 2.1.4. In this step, each color class is replaced by $8 \log D(D \log D)^{1/B}/B$ color classes. Over all the three steps of refinement, the number of color classes created is

$$\begin{aligned}&(2^{O(\log^*(C/D))}C/D) \cdot (32D/\log D) \cdot (8 \log D(D \log D)^{1/B}/B) \\ &= C(D \log D)^{1/B}2^{O(\log^*(C/D))}/B.\end{aligned}$$

This gives a schedule of length at most $(L+D)C(D \log D)^{1/B}2^{O(\log^*(C/D))}/B$.
□

2.2 Lower bounds on schedule length

We show that there exists a routing problem that requires at least $\Omega(LCD^{1/B}/B)$ time to route. This matches our upper bounds to within small factors. This proof is a generalization of the proof for $B = 1$ in [40]. The structure of our proof is similar to theirs.

Theorem 2.2.1 *There exists a routing problem with message-length L, congestion C, and dilation D such that the total time required to route all the messages is $\Omega(LCD^{1/B}/B)$, for any $C, D \geq B+1$, and for any $L = (1+\Omega(1))D$.*

Proof: The key idea is to construct a set of messages such that *every* set of $B+1$ messages passes through some specific edge. Intuitively, this implies that if the messages are long enough, at most B messages can make "progress" at any time-step (this is made precise later). This gives the required bound.

Our construction is as follows. We start out with a set of M' messages, where M' is an integer such that

$$2\binom{M'-1}{B} - 1 \leq D < 2\binom{M'}{B} - 1$$

We construct a network using two types of edges: primary edges and secondary edges. The network has a total of $\binom{M'}{B+1}$ primary edges – every set of $B+1$ messages passes through a distinct primary edge. Further, the end-point of each primary edge is connected to the end-points of every other primary edge using secondary edges. 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.

The congestion and dilation of our construction are evaluated as follows. It is clear that each message passes through $\binom{M'-1}{B}$ primary edges and $\binom{M'-1}{B} - 1$ secondary edges, for a total dilation of $2\binom{M'-1}{B} - 1$. This is at most D by our choice of M'. We could make it exactly D by adding extra edges at the end of the path for each message, if necessary. The congestion of each primary edge is clearly $B+1$. The net congestion is also $B+1$, since the secondary edges have congestion at most B. To obtain a set of messages with congestion C, we simply replicate each message $C/(B+1)$ times so as to obtain a total of $M = CM'/(B+1)$ messages.

We now evaluate the time to route the M messages. A message is defined to make progress in a specific time-step if and only if the message moves in that time-step and one of its first $L-D$ flits reaches its destination. Note that a message making progress must occupy every edge on its path. Since every set of $B+1$ messages pass through an edge, only B messages can make progress at each time step. Therefore, the time required to route all messages is at least $(L-D)M/B$. Since $L-D = \Omega(L)$ and since $M' = \Omega(BD^{1/B})$, the time taken is $\Omega(LCD^{1/B}/B)$. $\square$

3 Routing on butterfly networks

In Section 3.1, we provide a wormhole routing algorithm for the butterfly network. We show that an arbitrary q-relation can be routed on an n-input butterfly in time $O(L(q+\log n)(\log^{1/B} n)w_1(n,q)/B)$ flit steps, where $w_1(n,q)$ is a slowly-growing function of n and q, and $1 \leq B \leq \log\log n/\log\log\log n$. As a typical example, when $L = q = \Theta(\log n)$, the algorithm routes all the messages in time $O((\log^{2+1/B} n)w_1(n,q)/B)$, where $w_1(n,q) = O(\log\log n)$. This matches the best known results for $B = 1$ (to within the $w_1(n,q)$ factor), and is faster by a factor of about $B\log^{1-1/B} n$ for $1 < B \leq \log\log n/\log\log\log n$. The results in Section 3.1 also apply to the random routing problem with q 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 q messages per input requires $\Omega(Lql^{1/B}w_2^{-1}(n,q)/B)$, where $l = \min\{L, \log n\}$ and $w_2(n,q)$ is a slowly-growing function of n and q. As a typical example, when $L = q = O(\log n)$, the the lower bound evaluates to $\Omega((\log^{2+1/B} n)w_2^{-1}(n,q)/B)$, where $w_2(n,q) = O(\log\log n)$. Thus this matches the upper bound within a small poly-loglog factor. The results in Section 3.2 also apply to routing a random q-relation.

3.1 Butterfly algorithm

The algorithm for routing a q-relation on an n-input butterfly network proceeds in rounds. There will be a total of $2\log\log(nq)+1$ rounds. Each round

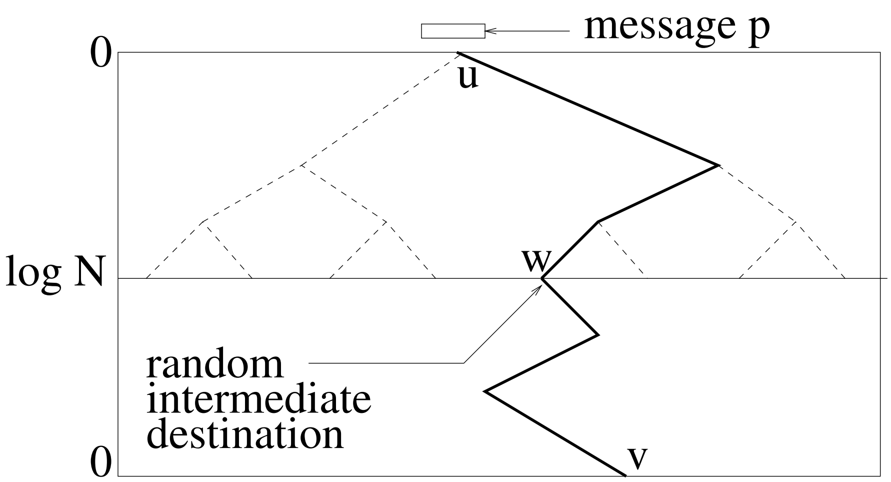

Figure 2: Routing message p using two passes through the butterfly

consists of the following steps (R takes on values from 0 to $2 \log \log(nq)$ 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 $\Delta = \beta q \log^{\frac{1}{B}} n/B$, 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 qn messages to reach their respective destinations.

Theorem 3.1.1 *The above algorithm delivers a copy of each message in a q-relation routing problem in*

$$O(L(q + \log n)(\log^{1/B} n)w_1(n, q)/B)$$

flit steps, with high probability, provided $B \leq \log\log n/\log\log\log n$*, where* L *is the number of flits per message,* B *is the number of virtual channels, and* $w_1(n, q) = \log\log(nq)$*.*

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

Invariant 3.1.2 *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* q*, with high probability.*

We show that Invariant 3.1.2 holds by using induction on the number of rounds. Clearly, Invariant 3.1.2 holds at the beginning of round 0, since each input has q messages and each output is the destination for q messages. Using the following lemmas, we show in Theorem 3.1.6 that if the Invariant 3.1.2 holds in round i, it continues to hold in round $i + 1$.

Lemma 3.1.3 *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 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 B 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 B 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 B 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 3.1.2, a total of $2^{l_i}q$ messages could use edge e_i. Therefore, the total number of ways of choosing B-tuples of messages that could pass through e_i is $\begin{pmatrix} 2^{l_i}q \\ B \end{pmatrix}$. Therefore, the total number of ways of choosing B-tuples of messages for all the edges $e_i, 1 \leq i \leq t$, is at most

$$\prod_i \begin{pmatrix} 2^{l_i}q \\ B \end{pmatrix} \leq (qe/B)^{Bt} \prod_i 2^{l_i B}$$

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

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

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

$$(qe/B)^{Bt} \cdot \prod_i 2^{l_i B} \cdot \prod_i 2^{-(l_i+1)B} \cdot \frac{1}{\Delta^{Bt}} \leq (\frac{qe}{2\Delta B})^{Bt}$$

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

$$(2\log n)^t (\frac{qe}{2\Delta B})^{Bt} \leq \frac{2^t}{(2\beta/e)^{Bt}} \leq \frac{1}{2^{12t}}$$

for a sufficiently large value of β. $\Box$

Lemma 3.1.4 *Let each input (resp. output) have q messages in the beginning of a round. With high probability, there are at least $\frac{3}{4}q$ messages in each input (resp. output) with distinct colors.*

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

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

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

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

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

Lemma 3.1.5 *Assume that Invariant 3.1.2 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 $q/2$, with high probability.*

Proof: Let each input (resp. output) have q messages in the beginning of a round. Using Lemma 3.1.4, at least $\frac{3}{4}q$ of the messages in each input (resp. output) pick different colors, with high probability. We claim that, with high probability, at least $\frac{q}{2}$ of the messages that picked different colors succeed in reaching their destinations. To prove this claim, we use Lemma 3.1.3 to bound the probability that there exists a subset of $\frac{q}{4}$ messages with different colors that fail to reach their destination. This probability is at most

$$\binom{3q/4}{q/4} \frac{1}{2^{12q/4}} \leq \frac{1}{2^{2q}}$$

which is polynomially-small, since $q \geq \log n$. □

Theorem 3.1.6 *If Invariant 3.1.2 holds in round i, it continues to hold in round $i+1$, with high probability.*

Proof: Using Lemma 3.1.5, we know that if the Invariant 3.1.2 holds at the beginning of round i, there are at most $q/2$ 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 q. Thus Invariant 3.1.2 holds in round $i+1$. □

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

Proof of Theorem 3.1.1: First, we show that at least one copy of each message succeeds in reaching its destination after the last round, 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 $2^{2\log\log(nq)} = \log^2(nq)$ copies of p, of which at least $\log(nq)$ copies will receive different colors, with high probability (This follows from an argument similar to the proof of Lemma 3.1.4). The probability that none of the $\log(nq)$ copies gets through in last round is at most

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

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

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

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 L flit 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 L steps. There are $\Delta = \beta q \log^{\frac{1}{B}} n/B$ sub-rounds in each round. Therefore, each round takes $L\beta q \log^{\frac{1}{B}} n/B + 2\log n$ flit steps, which is $O(Lq\log^{\frac{1}{B}} n/B)$ flit steps. Since there are $2\log\log(nq)$ rounds, the total time taken is $O(Lq\log^{\frac{1}{B}} n \log\log(nq)/B)$ which is $O(Lq(\log^{\frac{1}{B}} n)w_1(n,q)/B)$ flit steps.

This algorithm can be used to route messages even when $q < \log n$. We simply duplicate packets such that $\Theta(\log n)$ messages originate at each input and $\Theta(\log n)$ messages are destined for each output. The analysis is identical except that q must be replaced by $\log n$ in the expression for the running time. $\square$

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 q 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 q-relation.

The proof of this lower bound generalizes the lower bound for $B = 1$ presented by Ranade, Schleimer, and Wilkerson[40]. The structure of our proof follows very closely the structure of theirs. Note that the lower bound matches our upper bound to within small factors.

Theorem 3.2.1 *With high probability, any one-pass butterfly routing algorithm must take time T to route a random routing problem with q messages per input, where*

$$T = \Omega(Lql^{\frac{1}{B}} w_2^{-1}(n, q)/B)$$

where $l = \min\{L, \log n\}$, and $w_2(n, q)$ is a slowly-growing function of n and q.

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 $l = \min\{L, \log n\}$. 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 3.2.2 *A set of s messages is said to collide if there exist $B + 1$ messages in the set whose paths all use a single edge in the truncated-butterfly.*

The proof of Theorem 3.2.1 proceeds in two steps. First, we show that *every* set of $s = 3Bn \log^{2/B}(q \log n)/l^{\frac{1}{B+1}}$ messages chosen from the total of nq messages collide, with high probability (Theorem 3.2.5). Next, in Theorem 3.2.6, we show that if the routing algorithm completes in time at most $T = nqL/s$ then there exists a set of s messages that do not collide – which, using Theorem 3.2.5 is a low probability occurrence. This implies the result of Theorem 3.2.1.

We need the following lemmas to prove Theorem 3.2.5.

Lemma 3.2.3 *Let m balls be thrown independently and randomly into n bins, where $m \leq n$. The probability that no bin receives more than B balls is at most*

$$\exp\left\{-\frac{\alpha m^{B+2}}{(2Bn)^{B+1}B}\right\}$$

where α is a positive constant.

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

$$P = P_1 \cdot P_2 \cdot P_n \tag{1}$$

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

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

$$\begin{aligned} & \sum_{i \geq B+1} \binom{m/2}{i} \frac{1}{n^i} \left(1 - \frac{1}{n}\right)^{m/2-i} \\ \geq & \binom{m/2}{B+1} \frac{1}{n^{B+1}} \left(1 - \frac{1}{n}\right)^{m/2} \\ \geq & \alpha' \frac{m^{B+1}}{(2Bn)^{B+1}} \end{aligned}$$

where α' is a positive constant. Thus, for all $i \leq m/2B$,

$$P_i \leq 1 - \alpha' \frac{m^{B+1}}{(2Bn)^{B+1}} \tag{2}$$

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

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

where α is a positive constant.

Lemma 3.2.4 *Given an arbitrary set of*

$$s = 3Bn \log^{2/B}(q \log n)/l^{\frac{1}{B+1}}$$

messages, the probability that these messages do not collide is at most

$$\exp\left\{-\frac{\alpha s^{B+2} l}{(2Bn)^{B+1} B \log\log n}\right\}$$

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

First, we bound the probability that the s messages do not collide in a specific strip S. Strip S consists of n/m m-input sub-butterflies. Let s_i, $1 \le i \le n/m$, 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 s_i messages fail to collide at the outputs of the sub-butterfly can be upper bounded, using Lemma 3.2.3, to be at most

$$\exp\left\{-\frac{\alpha s_i^{B+2}}{(2Bm)^{B+1}B}\right\}$$

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

$$\prod_{i=1}^{i=n/m} \exp\left\{-\frac{\alpha s_i^{B+2}}{(2Bm)^{B+1}B}\right\}.$$

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

$$\exp\left\{-\frac{\alpha(sm/n)^{B+2}}{(2Bm)^{B+1}B}\frac{n}{m}\right\} = \exp\left\{-\frac{\alpha s^{B+2}}{(2Bn)^{B+1}B}\right\}.$$

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 sm/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 $l/\log m$ strips. Therefore, the probability that there is no collision in the entire truncated butterfly is at most

$$\exp\left\{-\frac{\alpha s^{B+2} l}{(2Bn)^{B+1} B \log\log n}\right\}.$$

$\square$

Theorem 3.2.5 *Every set of s messages chosen from the total of nq messages collides, with high probability, where $s = 3Bn\log^{2/B}(q\log n)/l^{\frac{1}{B+1}}$.*

Proof: There a total of $\binom{nq}{s}$ choices for the s messages. Thus, using Lemma 3.2.4, the probability that there exists a set of s messages that do not collide is at most

$$\begin{aligned}
& \exp\left\{-\frac{\alpha s^{B+2} l}{(2Bn)^{B+1} B \log\log n}\right\}\binom{nq}{s} \\
\leq\ & \exp\left\{-\frac{\alpha s^{B+2} l}{(2Bn)^{B+1} B \log\log n}\right\}\left(\frac{nqe}{s}\right)^{s} \\
=\ & \exp\left\{-s(\alpha\frac{s^{B+1} l}{(2Bn)^{B+1} B \log\log n} - \log(nqe/s))\right\} \\
\leq\ & \exp\left\{-s(\alpha\log^{1+2/B}(q\log n) - O(\log(q\log n)))\right\} \\
\leq\ & \exp\{-\omega(s)\} \leq \frac{1}{n^{\omega(1)}}
\end{aligned}$$

(Recall that $s = 3Bn\log^{2/B}(q\log n)/l^{\frac{1}{B+1}}$ and $l = \min\{L, \log n\}$.) Thus, the probability that there exists a set of s messages that do not collide is polynomially small. $\square$

Theorem 3.2.6 *If an algorithm takes T time to route a total of nq messages then there exists a set of nqL/T messages that do not collide.*

Proof: This lemma and its proof is similar to Lemma 4 of [40]. 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

$l + iL$. 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 $l + iL$ 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 L 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/L phases imply that at least one phase has nqL/T messages. $\square$

We can now prove the main lower bound theorem.

Proof of Theorem 3.2.1: From Theorem 3.2.5, we know that the probability that there exists a set of s messages that do not collide is small. Therefore, setting $s = nqL/T$, and using Theorem 3.2.6, the routing algorithm must take time

$$T = nqL/s = nqL \cdot (l^{\frac{1}{B+1}}/3Bn \log^{2/B}(q \log n)).$$

Simplifying further, we get

$$T \geq Lql^{\frac{1}{B}} w_2^{-1}(n, q)/B,$$

where $w_2(n, q) = O(l^{1/B^2} \log^{2/B}(q \log n))$. $\square$

4 Acknowledgments

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. The authors would also like to thank Ronald Greenberg for helpful comments.

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$^{\mathrm{TM}}$ 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 hardware help routing? In *Proceedings of the 25th Annual ACM Symposium on the Theory of Computing*, pages 573–582, 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] R. Cypher, F. Meyer auf der Heide, C. Scheideler, and B. Vöcking. Universal algorithms for store-and-forward and wormhole routing. In *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing*, pages 356–365, May 1996.

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

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

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

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

[18] 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.

[19] R. Greenberg and H.-C. Oh. Universal wormhole routing. In *Proceedings of the 5th IEEE Symposium on Parallel and Distributed Processing*, pages 56–63, December 1993.

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

[21] P. Kermani and L. Kleinrock. Virtual cut-through: a new computer communications switching technique. *Computer Networks*, 3(4):267–286, 1979.

[22] 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.

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

[24] 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.

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

[26] 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.

[27] 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.

[28] 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.

[29] 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*, volume 2, pages 555–563, January 1995.

[30] F. T. Leighton, B. M. Maggs, and A. W. Richa. Fast algorithms for finding O(congestion + dilation) packet routing schedules. Technical Report CMU–CS–96–152, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, July 1996. To appear in *Combinatorica*.

[31] 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.

[32] 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.

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

[34] 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.

[35] 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*, pages 291–302, March 1995.

[36] 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.

[37] Rafail Ostrovsky and Yuval Rabani. Universal $O(\text{congestion}+\text{dilation}+\log^{1+\epsilon} N)$ local control packet switching algorithms. pages 644–653.

[38] 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.

[39] Yuval Rabani and Éva Tardos. Distributed packet switching in arbitrary networks. pages 366–375.

[40] 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.

[41] Christian Scheideler. *Universal routing strategies for interconnection networks*, volume 1390 of *Lecture Notes in Computer Science*. Springer-Verlag Inc., New York, NY, USA, 1998.

[42] C. Scheideler and B. Vöcking. Universal continuous routing strategies. In *Proceedings of the 8th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 142–151, June 1996.

[43] 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.

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

[45] Aravind Srinivasan and Chung-Piaw Teo. A constant-factor approximation algorithm for packet routing, and balancing local vs. global criteria. pages 636–643.

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

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