Nearly Tight Bounds for Wormhole Routing

Abhiram Ranade* Saul Schleimer† Daniel Shawcross Wilkerson‡

University of California, Berkeley, California 94720

Abstract

We present nearly tight bounds for wormhole routing on Butterfly networks which indicate it is fundamentally different from store-and-forward packet routing. For instance, consider the problem of routing $N \log N$ (randomly generated) $\log N$ length messages from the inputs to the outputs of an N input Butterfly. We show that with high probability that this must take time at least $\Omega(\log^3 N/(\log\log N)^2)$. The best lower bound known earlier was $\Omega(\log^2 N)$, which is simply the flit congestion in each link. Thus our lower bound shows that wormhole routing (unlike store-and-forward-routing) is very ineffective in utilizing communication links. We also give a routing algorithm which nearly matches our lower bound. That is, we show that with high probability the time is $O(\log^3 N \log\log N)$, which improves upon the previous best bound of $O(\log^4 N)$. Our method also extends to other networks such as the two-dimensional mesh, where it is nearly optimal. Finally, we consider the problem of offline wormhole routing, where we give optimal algorithms for trees and multidimensional meshes.

Keywords: wormhole routing, packet routing, cut-through, off-line routing, butterfly, multi-dimensional mesh, trees.

1 Introduction

Wormhole routing is an extremely popular strategy for data movement in parallel computers and is used in a variety of machines such as Intel Delta, MIT J machine, MIT April and others. In wormhole routing, a message is transmitted as a contiguous sequence of flits (flow control units) and the sequence moves along the path from the source to the destination in a pipelined manner, like a burrowing worm. There two are defining characteristics of wormhole routing: (i) *message contiguity* Every edge along the path must transmit all flits of the message in a contiguous manner, i.e. the bits of two different messages cannot be interleaved. (ii) *minimal buffering* Each intermediate node can only buffer a few flits. (Thus, if the head of the message cannot move forward because another message is using the edge it wants, then it message must wait, and as it waits it occupies a contiguous sequence of edges along its path.) Both of these properties make for simple hardware implementations. For example, the bookkeeping at each node is simplified because bits of the message cannot be interleaved, and since the queues at each intermediate node are only required to buffer few flits, they can be made simple, small, and fast.

Most of the theoretical work in message routing is based on the much simpler *(store-and-forward) packet routing model.* Packet routing is effectively a special case of wormhole routing in that each message is assumed to consist of a single flit. Another model that is fairly well understood is the *cut-through model* which is identical to the wormhole routing model, except that intermediate nodes in the network are assumed to have enough storage to buffer several entire messages, not just several flits.

It is natural to divide the problem of message routing (in any of the three models) into two parts: (i) *Path selection,* in which each message is assigned a path along which it moves, and (ii) *Scheduling* the movement of messages along the path assigned to it. Path selection algorithms are typically designed to minimize two parameters: (i) *congestion c:* the maximum number of flit paths traversing any single link of the network, and (ii) *dilation d:* the length of the longest path that any message must travel. Each of these parameters is clearly a lower bound on the time required to route the given set of messages. The goal of the scheduling step is to compute a schedule for

*Division of Computer Science, ranade@cs.berkeley.edu

†saulsch@ocf.berkeley.edu

‡Department of Mathematics, wilkerso@math.berkeley.edu

This research is partially supported by NSF-DARPA grant CCR-9005448. This paper will be published in the proceedings of FOCS 1994.

moving messages and if possible match the inherent lower bounds, i.e. finish routing in $O(c+d)$ steps. It is necessary, of course, that the path selection and scheduling algorithms be distributed (i.e. run on the network itself) and on-line (the time to compute the paths and the movement schedule is also included in the total routing time).

This paper is concerned with the question of scheduling message movement in the wormhole routing model. The input to our problem is the communication network and a set of messages along with their paths. The specification of the paths determines the parameters c and d as defined above, and the goal is to schedule message movement so that routing finishes in time $O(c+d)$, if possible. In previous work, Felperin, Upfal and Raghavan showed that this bound can be matched on two dimensional meshes[2], and gave suboptimal upper bounds for the butterfly. Their results were extended by Greenberg and Oh[3] to levelled directed networks and fat-trees. Except for two-dimensional meshes, all these results were substantially worse than the lower bound of $\Omega(c+d)$.

The situation is considerably different in packet routing. In this case, Leighton, Maggs and Rao[6] show that schedules achieving time $O(c+d)$ exist. Leighton, Maggs, Ranade and Rao [5] give distributed, on-line algorithms to finish the routing in time $O(c+d)$ for worst-case as well as average case routing problems on most known networks.

1.1 Main results and Overview

In this paper we improve the upper bounds as well as the lower bound for the time to route messages on the Butterfly network, for the case in which the messages are reasonably long ($\log N$ flits or more). Our lower bounds show that for many natural Butterfly routing problems the routing time must be $\omega(c+d)$. This implies that wormhole routing is fundamentally different from packet routing.

In Section 2 we present a generic algorithm for wormhole routing on levelled directed networks. For reasonably long messages, our algorithm considerably improves upon the one presented by Greenberg and Oh[3].

Our Butterfly algorithm presented in Section 2.4 is an instance of our generic algorithm. We consider two problems on the N-input butterfly: in the *Lightly loaded* problem each input sends 1 message to a randomly destined output, and in the *Heavily loaded* problem each input sends $\log N$ randomly destined messages. In each case $L \geq \log N$ denotes the number of flits in each message. For the lightly loaded problem our algorithm takes time $O(L \log N \log\log N)$. In comparison, the algorithm of Felperin et al[2] takes time $O(L \log N \min(L, \log N))$. For $L = \log N$, our time bound is $O(\log^2 N \log\log N)$, as opposed to $O(\log^3 N)$ for Felperin et al.

The heavily loaded problem is the most interesting of the lower bounds presented (Section 3). For the heavily loaded problem, with $L = \log N$, we show that routing requires time $\Omega(\log^3 N / \log^2 \log N)$, even if offline computation is allowed for computing schedules. Notice that the congestion+dilation lower bound for this problem is only $\Omega(\log^2 N)$. Our algorithm from Section 2.4 completes routing in time $O(\log^3 N \log\log N)$, nearly matching the improved lower bound. Note that the previous upper bound for this problem, implied in [2] is $O(\log^4 N)$.

Our generic algorithm for levelled directed networks can also be used to construct an on-line routing algorithm for routing "one-bend paths" on two dimensional meshes. This algorithm (Section 2.4) matches the performance of the algorithm of Felperin et al[2] for large enough messages.

In Section 4 we present offline algorithms to construct schedules for constant dimensional meshes and trees. We present algorithms that route in the optimal time $O(c+d)$ on these networks. Our results on trees improve upon the ones by Bhatt et al [1], and extend the algorithm of [1] for two dimensional meshes to higher dimensions.

In Section 5 we present a wormhole routing problem for which the routing time is $\Omega(cd)$. Any wormhole routing problem can be solved in $O(cd)$ time using offline computation, and our example shows that this bound is existentially tight.

We begin by clarifying our model of wormhole routing.

1.2 Model

Our model is similar to that of Felperin et al[2]. A routing network is a directed graph, where processors and switches are nodes and channels are directed edges. We only consider networks with bounded degree. A *flit* (flow control unit) is the atomic unit of information that may move through the network. It takes unit time to cross any edge and no two flits may traverse the same edge at the same time. Each node has a single buffer capable of holding one flit for each incoming link.

A message consists of a sequence of flits. In this paper we consider fixed length messages, each having L flits. Once the head (first flit) of a message is transmitted over any link, all other flits must be transmitted

over it before the flits belonging to any other message can be transmitted. In each time step the head flit may advance along its path if that link is free, and the buffer unoccupied; else it remains in its current buffer. Following Felperin et al[2] and Greenberg and Oh[3] we assume that when the head of a message moves forward, all its flits also move forward.

2 A routing algorithm on levelled directed networks

A *d-levelled directed network* G is a network in which each vertex is assigned a level in the range 0 through d and such that each edge starts at a vertex in level i and ends at some vertex in level $i+1$ (for some i). A *routing problem* on the network is a (multi) set P of paths that we want the messages to traverse. Paths must start on level 0 and terminate on level d. M denotes the number of such paths, and L the length of each message. We assume that $L \geq d$.

A useful tool for analyzing a routing problem instance is the *path graph*.

Definition 1 *Consider a set of paths P in G. Define the* path graph of P *to be the graph $\mathcal{P} = (P, E)$ where for any two paths p and q in G $(p,q) \in E$ iff p and q share an edge in the network.*

We will use Δ to denote the degree of the path graph.

2.1 The greedy algorithm

We first describe the natural greedy algorithm and analyze it. The greedy algorithm is useful as a subroutine in our algorithm. As the name suggests, the algorithm is to move each message forward whenever possible. If two messages need to be transmitted along the same edge then one of them is chosen arbitrarily and the other waits until the edge is available. We present a simple estimate of the time required by the greedy algorithm. Our proof is similar to the idea in Lemma 1.8 of Leighton[4].

Lemma 1 *Consider a set of paths on a d-levelled directed network and let C be a connected component of the associated path graph. Using the greedy algorithm all the messages in C are delivered in time $d + |C|L$, where $|C|$ denotes the number of messages in C.*

Proof: We first show the following: if some level i contains a flit from C at time t, then every level $i, i+1, \ldots, \min(d,t)$ contains a flit from C at time t. Let t_0 be the smallest t for which the above assertion is false. Then at time t_0 there exists i such that level $i+1 \leq \min(d,t)$ does not contain a flit from C but level i does. But this flit must have been in levels $i-1$ or i at time step $t_0 - 1$. But since the assertion is true for step $t_0 - 1$, we know that levels $i, \ldots, \min(d, t_0 - 1)$ were occupied[1]. Thus some flit f_1 in level i did not move into level $i+1$ at step t_0. But this can only be due to the corresponding buffer being occupied by a flit f_2 that moved in at step t_0, or was already there and did not move during step $t_0 - 1$. In either case, we have shown that at time t_0 level $i+1$ must contain a flit from C.

Thus at least one flit from C gets delivered to level d at each timestep on or after step d. Then the lemma follows by noting that C contains only $|C|L$ flits. ■

2.2 Our algorithm

Our algorithm builds upon the "delayed greedy" idea of Felperin et al[2], and has two phases:

phase one Each message picks a color at random from $\{1 \ldots \alpha\Delta\}$, where α is constant which we will fix later.

We then run $\alpha\Delta$ sub-phases, such that in the i^{th} sub-phase only the worms of color i participate. In sub-phase i we run the greedy algorithm for $L\log\Delta + d$ routing steps on messages of color i. The messages that are not completely delivered by that time are removed from the network.

phase two We now disregard the previously assigned colors, and run the greedy algorithm on all messages that were undelivered in phase 1, until all of them are delivered.

Phase 2 requires each level 0 node to know which of its messages were delivered in phase 1 (and which need to be transmitted in phase 2). For $L \geq d$ this is easily done as follows. A message will be delivered during phase 1 if at step $L\log\Delta$ its tail is already injected into the network. This is because $L \geq d$ guarantees that it will then have already occupied all the links it needs, and thus can proceed unhindered to be completely delivered by step $L\log\Delta + d$.

[1] Note that all flits are on level 0 when routing begins and a flit is considered delivered when it has reached level d.

2.3 Time analysis

Let $\mathcal{P}_0$ be the path graph of the entire routing problem, and let $\mathcal{P}_1$ be the path graph of those messages that remain to be delivered at the end of phase one.

Theorem 1 *The algorithm delivers all messages in time $O(L(\Delta \log \Delta + \log MN) + d\Delta)$ with high probability.*

Proof: Phase one takes time $\alpha\Delta(L\log\Delta + d)$, by construction.

Lemmas 2 and 3 below will show that $\mathcal{P}_1$ has no connected components of size $\log MN$ with high probability. Thus by lemma 1 the time for phase 2 is at most $L \log MN + d$. ■

Lemma 2 *If there is a component C' in $\mathcal{P}_1$ of size at least u then there is a connected subgraph C in $\mathcal{P}_0$ of size at least u that consists of the union of monochromatic components each of size at least $\log\Delta$.*

Proof: C is simply the union of the collection of those monochromatic connected components of $\mathcal{P}_0$ that contain at least one of the messages in C'. Clearly C is connected in $\mathcal{P}_0$, and the number u of messages in C is at least u.

By construction, we know that each monochromatic component of C is not completely delivered during phase one. Thus by lemma 1, each of these components must have size at least $\log\Delta$. ■

Lemma 3 *Let C denote the largest connected subgraph of $\mathcal{P}_0$ that is a union of monochromatic components, each of size at least $\log\Delta$. With high probability, C has $O(\log MN)$ vertices.*

Proof: Let u denote the number of vertices in C, and let $c_1, c_2, \ldots, c_k$ denote the monochromatic components constituting C. By lemma 2 we know that $k \le u/\log\Delta$. Each c_i has a monochromatic spanning tree. These trees may be joined to form a colored spanning tree of C with k contiguous monochromatic subtrees. We will upperbound the probability of the existence of such a colored tree.

We first count the number of possible spanning trees of the form mentioned above.

1. There are at most M ways to fix u.
2. There are $u4^u$ ways to pick a rooted unlabeled tree on u nodes.
3. There are at most 2^u ways to mark k edges of this tree as chromatic boundaries. Incidentally, this marking fixes the value of k.
4. The edges selected above split the tree into subtrees, and we must choose a color for each subtree. This can be done in $(\alpha\Delta)^k$ ways. This is at most $(\alpha\Delta)^{u/\log\Delta} \le (2\sqrt{\alpha})^u$ ways.
5. We embed this tree into the path graph. There are M ways to embed the root. Each node is then embedded to a neighbor of its parent. There are Δ^{u-1} ways to do this, for a total of at most $M\Delta^u$ ways.

For this tree to actually arise, the colors of all of the nodes in the tree must agree with the colors picked in item 3 above. For any particular colored tree the probability that it actually occurs is $\le (\alpha\Delta)^{-u}$. Thus, the probability that there exists a tree of size u in $\mathcal{P}_0$ is at most:

$$\frac{Mu4^u 2^u (2\sqrt{\alpha})^u M\Delta^u}{(\alpha\Delta)^u} \le \left(\frac{32 \cdot 2^{2\log M/u}}{\sqrt{\alpha}}\right)^u$$

We will ensure that $u \ge 2\log M$, and choose $\alpha = (128)^2$. This will make the quantity inside parenthesis be at most $1/2$. Thus, choosing $u = 2\log M + \beta \log N$, we can force the probability to be smaller than $N^{-\beta}$, for any β. ■

We have not attempted to minimize α, but clearly, a much smaller value should also suffice.

2.4 Applications

We instantiate the generic algorithm described in the previous section for N-input Butterflies and N node square meshes. For butterflies, $d = \log N$ and $L \ge \log N$. For meshes, we use $d = 2\sqrt{N} - 2$ and $L \ge 2\sqrt{N} - 2$.

Lightly loaded butterfly Each input has one randomly destined message. Using Chernoff bounds, it is easy to show that with high probability there are $O(\log N)$ messages touching the path of any message, giving the degree Δ of the path graph to be $O(\log N)$.

Applying our upperbound, the routing time is bounded above by

$$O(L \log N \log\log N).$$

We note that this is nearly optimal. This follows just by considering the (flit) congestion, which gives a high probability lower bound of

$$\Omega\left(\frac{L\log N}{\log\log N}\right)$$

Heavily loaded butterfly In this, each input has $\log N$ messages, so that $M = N \log N$. Using Chernoff bounds it is possible to show that $O(\log N)$ messages pass through any link with high probability. Since each message traverses $\log N$ links, we have the path graph degree $\Delta = O(\log^2 N)$. The routing time is then bounded above by

$$O(L \log^2 N \log\log N).$$

This time is nearly optimal, since using Theorem 2 for $L \geq \log N$, this problem has a lower bound of

$$\Omega\left(\frac{L \log^2 N}{\log^2 \log N}\right)$$

The mesh We consider the problem in which each node in the mesh sends out a single message to a randomly chosen node. We only consider how to route messages going "in the north-east" direction, other directions can be handled by making four copies of the mesh[5]. We see the mesh as a $(2\sqrt{N} - 2)$-levelled directed network where the north-west to south-east diagonals form the levels. The mesh and the paths on it are then extended in the natural way so that all paths start and end on the same level. Using Chernoff bounds it is easily seen that $O(\sqrt{N})$ messages share edges with any message, giving $\Delta = \sqrt{N}$ for this problem. This gives a time bound of

$$O\left(L\sqrt{N} \log N\right).$$

The time for higher loading increases proportionally. The best lower bound for this problem is based on flit congestion, and is only $\Omega(L\sqrt{N})$. Felperin et al give an algorithm that uses time $O(L\sqrt{N})$, but they require more complicated paths. For one bend paths their algorithm and analysis (specialized for the mesh) gives the same time as ours, though their algorithm works well with slightly shorter messages.

3 Butterfly lowerbound

Consider routing problems on an N-input Butterfly in which each input has $\log N$ messages destined to uniformly randomly and independently chosen outputs. Let $M = N \log N$ denote the total number of messages, and L denote the length of each message. We will prove lower bounds on the time to route such problems.

Using Chernoff bounds (e.g. see Leighton[4]) it is possible to show that the congestion in every link will be $L \log N$ with high probability.

Nevertheless it turns out that with high probability when $M = N \log N$ the problem requires $\Omega\left(\frac{Ll \log N}{8 \log^2 \log N}\right)$ time, where $l = \min\{L, \log N\}$. In particular it is interesting to remark that our result is stronger than the congestion bound as long as $L > 8 \log^2 \log N$.

We begin the proof by reducing the problem to another that is simpler to analyze.

Definition 2 *The* N-input truncated butterfly *is the network consisting of only the first l levels of the N-input butterfly.*

Clearly, any routing problem on the butterfly restricts naturally to a routing problem on the truncated butterfly, and any butterfly routing algorithm can be used in the obvious way to route a given problem on the truncated butterfly in the same or better time as on the butterfly. We prove a lowerbound on truncated-butterfly routing times, and thus butterfly routing times.

Lemma 4 *Without loss of generality we may assume that message movement on the truncated butterfly occurs in phases of length L. That is, if the routing begins at time 0, all message heads arrive at time $l + iL$ for some non-negative integer i.*

Proof: Suppose not. Consider the first message m_1 whose arrival time is not in phase. Without loss of generality, it was blocked at some point by another message. (Otherwise we could have released m_1 earlier). Let m_2 be the last message that blocked it. When the tail of m_2 passes the head of m_1, m_1 moves forward, always remaining on the level immediately behind that of the tail of m_2, arriving at the end of the network L steps after the head of m_2. This contradicts the assumption that m_1 was the first message whose arrival time was out of phase. ■

Theorem 2 *With high probability a randomly chosen problem requires time $\Omega\left(\frac{Ll \log N}{8 \log^2 \log N}\right)$, for $M = N \log N$.*

Proof: Notice that messages in the same phase cannot conflict. That is, the phases of the routing partition the M vertices of the path graph (of the truncated paths) $\mathcal{P}$ into independent sets. If the routing takes time T, then the number of parts is T/L and by an averaging argument one partition is of size at least ML/T. Lemma 5 below will show that any fixed subset of the vertices of $\mathcal{P}$ of size $B = \frac{ML}{T} = \frac{8N \log^2 \log N}{l}$

has probability at most $\exp\left(\frac{-lB^2}{4N\log\log N}\right)$ of being independent.

Since there are $\binom{M}{B}$ possible such subsets, by the union bound, the probability that one of them is independent is at most

$$\begin{aligned}
&\exp\left(\frac{-lB^2}{4N\log\log N}\right)\binom{M}{B} \\
&\quad\le \exp(-2B\log\log N)\left(\frac{Me}{B}\right)^B \\
&\quad= \left(\frac{Me}{B\log^2 N}\right)^B \\
&\quad= \left(\frac{le}{\log N\log\log N}\right)^B \\
&\quad= N^{-\omega(1)}.
\end{aligned}$$

■

Lemma 5 *If a multi-set of $B \ge 2N/\log N$ paths are chosen uniformly and independently at random in the truncated N-input butterfly then the probability that they are independent is at most*

$$\exp\left(\frac{-lB^2}{4N\log\log N}\right).$$

Proof: Let $n = \log N$. We first consider the probability that the B messages conflict at level $\log n$ of the truncated butterfly. Consider the subgraph of the truncated butterfly consisting only of the first $\log n$ levels. It is easily seen that this consists of N/n small butterflies, each having n inputs. Assume that the source processors for the B messages are chosen, and that this results in B_i messages being placed in the ith small butterfly, $i = 1, \ldots, N/n$. We estimate the probability that there are no collisions in any of the small butterflies at the $\log n^{th}$ level.

Consider the paths of the B_i messages inside the i^{th} small butterfly. There are n^{B_i} total ways in which the right endpoints of the paths can be chosen, of which only $n(n-1)\ldots(n-B_i+1)$ ways do not collide at the last level. Thus the probability of not having collisions at the $\log n^{th}$ level is

$$\begin{aligned}
&1\left(1-\frac{1}{n}\right)\left(1-\frac{2}{n}\right)\ldots\left(1-\frac{B_i-1}{n}\right) \\
&\quad\le \exp\left(-\left(\frac{1}{n}+\frac{2}{n}+\ldots+\frac{B_i-1}{n}\right)\right) \\
&\quad= \exp\left(\frac{-B_i(B_i-1)}{2n}\right).
\end{aligned}$$

The probability that this happens in every small butterfly is

$$\begin{aligned}
&\prod_{i=1}^{N/n}\exp\left(\frac{-B_i(B_i-1)}{2n}\right) \\
&\quad= \exp\left(\frac{1}{2n}\left(B-\sum B_i^2\right)\right) \\
&\quad\le \exp\left(\frac{1}{2n}\left(B-\sum\left(\frac{B}{N/n}\right)^2\right)\right) \\
&\quad= \exp\left(\frac{B}{2}\left(1/n-B/N\right)\right) \\
&\quad\le \exp\left(\frac{-B^2}{4N}\right)
\end{aligned}$$

using $B \ge 2N/n$. Notice that to obtain this we used no information regarding the start positions of the paths at level 0. Thus the argument can be repeated to obtain an upperbound on the probability that there is no collision on level $2\log n$, and indeed $j\log n$ for any $j = 1, \ldots, \frac{l}{\log n}$. In other words, given that there has not been any collision at level $\log n$, the probability that there is no collision at level $2\log n$ is also given by the above expression. Again since no information was used regarding the initial positions of the paths in the small butterflies, we may multiply these probabilities to obtain an upperbound on the probability that there are no collisions on any level of the form $j\log n$ (Which of course is also an upperbound on the probability that there are any collisions at all). This gives

$$\left(\exp\left(\frac{-B^2}{4N}\right)\right)^{\frac{l}{\log n}} = \exp\left(\frac{-lB^2}{4N\log\log N}\right).$$

■

4 Optimal offline algorithms for multi-dimensional meshes and trees

We show how to compute offline routing schedules for sets of paths with flit congestion c and dilation d on trees and constant dimensional meshes. The schedules have length $O(c+d)$ and thus match the congestion+dilation lower bound. We require on k dimensional meshes that the paths consist of at most one straight segment in each dimension. Our tree results improve the ones in [1], while our k dimensional mesh results generalize the case $k = 2$ presented there.

The schedules we compute actually satisfy a stronger model, the *bufferless routing* model[1]. In this model we have the additional restriction that once any message starts moving it does not stop along the way. Clearly, every bufferless schedule is a wormhole schedule, but not vice versa.

We will first derive bufferless schedules for problems where the message length $L = 1$. We will then show (Section 4.5) how to generalize the method to give optimal schedules for general L. Let c_L denote the flit congestion for length-L messages. Clearly $c_L = Lc_1$. In particular, we will show that if $T(L)$ denotes the time for scheduling length-L paths, then we will have

$$T(1) = O(c_1 + d) \Longrightarrow T(L) = O(c_L + d)$$

for all of the problems we consider.

4.1 Problem statement

A *routing problem* consists of a graph $G = (V, E)$ and a set P of paths on G. c denotes the maximum number of paths passing through any edge and d the length of the longest path.

Define a *coloring* of the paths P to be a map $C : P \rightarrow \boldsymbol{N}$. Let p, q be paths which share some edge e. Let $d_p(e)$ denote the distance along p from the initial point of p to the initial point of e. We say that paths p and q *conflict* at e if

$$d_p(e) + C(p) = d_q(e) + C(q).$$

We say a coloring is *good* if no two paths conflict.

Clearly coloring one path p can make some colors unavailable for another path q. (That is, the use of those colors for q would produce a conflict with p.) We say that a routing problem is *simple* if coloring p can make at most one color unavailable for each other path q. The mesh and tree problems we address here are clearly simple.

Notice that if we interpret the color of a path as a starting time for a message travelling that path in a routing schedule, then the good colorings are exactly the legal bufferless routing solutions. Thus our goal is to find a coloring that does not have any conflicts and that uses colors in the smallest range possible.

4.2 Algorithm

Our coloring algorithm uses a generic greedy strategy as follows:

1. Choose a total order $<^*$ on the paths, P. (The way that this is done distinguishes different versions of the algorithm for different types of networks, such and trees and k-dimensional meshes.) Initialize all paths to have no color.

2. Consider the paths in increasing order by $<^*$ and assign each path the smallest possible color such that there are no conflicts with paths that have already been colored.

The key question is how to find a total order $<^*$ that minimizes the maximum color required.

Definition 3 *A total order $<^*$ on paths is* k-entrant *if it is possible to associate with each path p a set* entrance(p) *of edges lying on p such that (i)* entrance(p) *contains at most k edges and (ii) if $p <^* q$ and p and q intersect, then some edge in* entrance(q) *lies on p.*

Lemma 6 *If P is a simple routing problem and there exists a k-entrant total order $<^*$ on P then there is a good coloring of P using colors only up to $k(c-1)+1$. (Where c is the message congestion as well as the flit congestion of P, since $L = 1$.)*

Proof: Use the greedy algorithm above. Consider what happens when we color a path q. Every path that (i) has already been assigned a color and (ii) intersects q passes through some edge in entrance(q).

There can be at most $k(c-1)$ such messages, since the congestion of each edge is at most c and since there are at most k vertices in entrance(q). Thus when we color q, there are at most $k(c-1)$ colors unavailable for use. Thus we may choose a color that is at most $k(c-1)+1$. ■

We will find 2-entrant total orders for trees and k-entrant total orders for k-dimensional meshes. Since k is a constant, the largest color used will be $O(c)$ and thus every message will be delivered in time $O(c + d)$.

4.3 The tree

Suppose G is a (bi-directed) tree. We can see that any set of paths is 2-entrant as follows. Pick some node to be the root and for any path p let its *highest point* be the vertex on p closest to the root. Again for any p let entrance(p) be the two edges in p adjacent to the highest point of p. Say $p <^* q$ iff the highest point of p is higher than that of q, breaking ties arbitrarily.

4.4 k-dimensional meshes

We only allow paths in which there are k segments and no two traverse the same dimension. Partition the paths into $2^k k!$ parts, one part for each possible permutation and orientation of the dimensions. We assign a path p to the part corresponding to the direction and orientation in which it traverses the dimensions. Each of these parts can be scheduled separately and then combined into one routing. (Since k is a constant, this does not change the order of the solution time.) From now on we restrict our attention to those paths that traverse the dimensions in increasing order and in the positive direction.

For any path p let entrance(p) consist of the very first edge of p and each edge following a corner vertex of p. (If a dimension is not traversed, then we include no edge for it.) There are at most k such edges. Define $<^*$ to be the reverse-lexicographic ordering on the coordinate vector of the initial points of the paths.

Lemma 7 *$<^*$ as defined above is k-entrant.*

Proof: Suppose paths p and q intersect along an edge. Follow both paths backward from that edge until the paths separate. Let e be their first common edge, and suppose that it traversed dimension j. For the paths to separate, one of them must have just reached a corner point or its initial point. Assume without loss of generality that it was q. Thus $e \in$ entrance(q) and we have an edge from entrance(q) lying on p.

Now consider the initial points $\langle q_1, \ldots, q_k \rangle$ and $\langle p_1, \ldots, p_k \rangle$ of q and p respectively. Since p and q intersect at a point before both of them have traversed any dimensions higher than j, for all $i > j$, $q_i = p_i$. Recall that as we walked backwards on q and p together, q turned off of dimension j first. Thus $p_j < q_j$. From this we see that $p <^* q$. ■

Thus the total number of colors necessary to color all of the paths is at most $2^k k!(k(c-1)+1) = O(c)$.

4.5 Multiflit messages

Notice that we could deal with messages that are L flits long by replacing each path with L paths (all with the same source and destination) and using the previous algorithm. The schedule thus obtained will use $O(c_L)$ colors, where c_L is the flit congestion of any edge – however the flits belonging to each message might not be scheduled in a contiguous manner.

We modify the greedy algorithm to ensure that all flits are scheduled in a contiguous manner. This means that in step 2 of the algorithm of section 4.2, we must find a *contiguous* sequence of L available colors to assign to the flits constituting the message (we color them all at once). Note however that since the flits of previous messages have also been colored contiguously, the unavailable colors occur in contiguous blocks of length L.

Notice then that any placement of $k(c-1)$ blocks of L unavailable colors must leave a block of L available colors somewhere in the range $1 \ldots (2L-1)k(c-1)+L$. Thus the maximum color that will be used is $(2L-1)k(c-1)+L = O(c_L)$. Thus the maximum delivery time will be $O(c_L + d)$.

5 A Problem requiring $\Omega(cd)$ steps

If d is the maximum path length, then clearly any order is d-entrant, for any routing problem. Thus in general, any routing problem can be offline scheduled to finish in $O(cd)$ time. Here we show a routing problem for which $\Omega(cd)$ is necessary, even with off-line scheduling.

Let p be a prime number. Let our levelled directed network consist of $2p+1$ levels (numbered 0 to $2p$) with p vertices each. Vertices in odd levels connect to the corresponding vertices in the next level by a single edge. Vertices in even levels connect to the next level by a copy of $K_{p,p}$.

We use p^2 messages of length $L \geq 2p+1$. Denote them $w_0 \ldots w_{p^2-1}$. Note that the path for each message w_i can be represented as a vector $v_i \in \{0 \ldots p-1\}^p$, where $v_{i,k}$ is the position of w_i on levels $2k$ and $2k+1$. Notice also that for any w_i and w_j, if there is a k such that $w_{i,k} = w_{j,k}$ then the paths for w_i and w_j collide along the edge from level $2k$ to level $2k+1$.

For any integer i, let $i_a = \left\lfloor \frac{i}{p} \right\rfloor$ and $i_b = i \bmod p$. We now define the paths of the worms. Let worm w_i travel the path given by the following vector.

$$v_{i,k} = \begin{cases} i_a & \text{if } k = 0 \\ i_b + k \cdot i_a & \text{otherwise} \end{cases}$$

Note that the maximum number of messages through any edge is p, so the flit congestion $c = pL$. The length of each path is $2p$, i.e. $d = 2p$.

We will show that every pair of paths collide in some edge. Consider w_i and w_j. If $i_a = j_a$, then they collide on the edge from level 0 to level 1. Otherwise,

$$i_b + k \cdot i_a \equiv j_b + k \cdot j_a \pmod{p}$$

has the solution

$$k_0 \equiv \frac{j_b - i_b}{i_a - j_a} \pmod{p}$$

since $\boldsymbol{Z}_p$ is a field. Thus paths w_i and w_j collide on the edge from level $2k_0$ to $2k_0 + 1$.

Notice that by a proof similar to that of lemma 4 in the butterfly lowerbound section we may assume that the messages will be routed in phases, where all worms in a phase arrive at the same time and arrivals are separated by L timesteps. But since all paths intersect and the length of each message is at least the length of the network, no two messages can be delivered at the same time! Therefore the total time is at least $p^2 L = \Omega(cd)$.

6 Open questions

6.1 Toward a possible extension of the butterfly upperbound for $L < d$.

The biggest unresolved question relates to short messages. Can we adapt the algorithm presented in Section 2 to messages with $L < d$? The main obstacle is how to inform level 0 nodes which of their messages got delivered in phase 1. The mechanism presented in Section 2 crucially exploits the assumption $L \geq d$. If we are allowed to run the network in reverse (possible if each node keeps a transcript of how it moved messages going forward) we can simply return each undelivered message back to its sender. By using minor additional features, we can even pipeline sub-phases of phase 1, so that the time for routing on levelled networks would become $O(L(\Delta \log \Delta + \log N) + d)$, and the routing times mentioned in Section 2.4 will be correct for any length messages. These times compare favorably with the previous work[2, 3] for $L > \log \log N$. But getting these timings requires extending the basic wormhole model as suggested above, and this seems to us to violate the basic motivation of the wormhole model: simplicity.

6.2 Other problems

The work presented here raises several questions.

A related open problem is to unify the algorithms known for wormhole routing and packet routing and produce a single algorithm and a single analysis for both. Packet routing, after all, is wormhole routing with $L = 1$. This we feel is a harder problem, since we do not know how to solve the heavily loaded packet routing problem without invoking random priorities as in [5][2] One approach to achieve this goal may be to strengthen Lemma 1. This lemma gives a really outrageous bound on the performance of the greedy algorithm, while extensive experiments suggest that the greedy algorithm performs very well.

There are open problems for long messages as well. It would be useful to tighten the gap between the Butterfly upper bound and the lower bound: for $L = \log N$, we conjecture the complexity of the lightly loaded problem is $\theta(\log^2 N)$, and $\theta(\log^3 N)$ for the heavily loaded problem. It would also be desirable to improve the analysis of the algorithm for the two dimensional mesh using one bend paths; or get better lower bounds.

References

[1] S. Bhatt, G. Bilardi, G. Pucci, A. Ranade, A. Rosenberg, and E. Schwabe. On Bufferless Routing of Variable Length Messages in Leveled Networks. In *First European Symposium on Algorithms*, pages 300–343, September 1992.

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

[3] R. Greenberg and H.-C. Oh. Universal Wormhole Routing. In *IEEE Symposium on Parallel and Distributed Processing*, 1993.

[4] F. T. Leighton. *Introduction to parallel algorithms and architectures.* Morgan-Kaufman, 1991.

[5] Tom Leighton, Bruce Maggs, Abhiram Ranade, and Satish Rao. Routing and Sorting on Fixed-Connection Networks. *Journal of Algorithms*, 16(4), July 1994.

[6] Tom Leighton, Bruce Maggs, and Satish Rao. Universal Packet Routing Algorithms. In *Proceedings of the IEEE Annual Symposium on The Foundations of Computer Science*, 1988.

[2] Maggs and Sitaraman have recently solved the light loading case.