Hard-Potato Routing

Costas Busch*
Brown University
Providence, Rhode Island
cb@cs.brown.edu

Maurice Herlihy†
Brown University
Providence, Rhode Island
herlihy@cs.brown.edu

Roger Wattenhofer‡
Brown University
Providence, Rhode Island
roger@cs.brown.edu

ABSTRACT

We present the first hot-potato routing algorithm for the $n \times n$ mesh whose running time on any "hard" (i.e., $\Omega(n)$) "many-to-one" batch routing problem is, with high probability, within a polylogarithmic factor of optimal. For any instance I of a batch routing problem, there exists a well-known lower bound LB_I based on maximum path length and maximum congestion. If LB_I is $\Omega(n)$, our algorithm solves I with high probability in time $O(LB_I \cdot \log^3 n)$. The algorithm is distributed and greedy, and it makes use of a new routing technique based on *multi-bend* paths, a departure from paths using a constant number of bends used in prior hot-potato algorithms.

1. INTRODUCTION

A *hot-potato* (or *deflection*) routing algorithm is a packet routing algorithm in which nodes have no buffers to store packets in transit: any packet that arrives at a node other than its destination must immediately be forwarded to another node. Hot-potato routing algorithms have been observed to work well in practice and have been used in parallel machines such as the HEP multiprocessor [17], the Connection machine [10], and the Caltech Mosaic C [16], as well as high speed communication networks [14]. Hot-potato routing algorithms are well-suited for optical networks [1; 9; 14; 19; 20] because it is difficult to buffer optical messages.

The network we consider here is the two-dimensional $n \times n$ mesh, one of the simplest networks for multiprocessor architectures. The nodes in the network are *synchronized*, namely, time is discrete and in each time step a node receives packets from its adjacent nodes, then makes routing decisions, and then forwards the packets to the adjacent nodes according to the routing decisions. At each time step a node is allowed to send at most one packet per link.

*Supported by NSF grant DMS-9505949.

†Supported by NSF grant DMS-9505949.

‡Supported by the Swiss National Science Foundation.

We consider "many-to-one" *batch routing problems.* In a batch routing problem each node is the source of exactly one packet at time zero. We are interested in finding how much time is needed until all the n^2 packets reach their destinations. Batch routing problems are "many-to-one" in the sense that a node may be the destination of multiple packets.

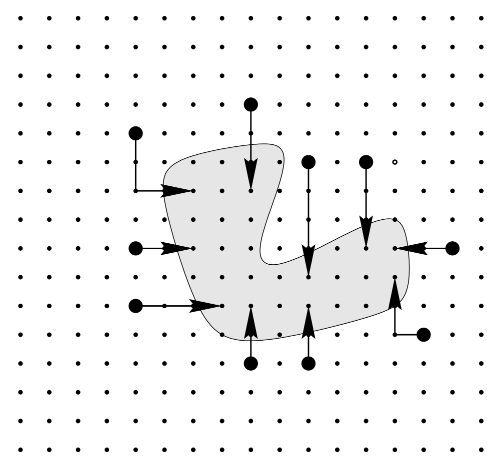

Figure 1: Packets with destinations in a region

Mansour and Patt-Shamir [13] have noted that there is a trivial lower bound for problems of this kind. If a packet's source and destination are separated by distance d, then no routing algorithm can deliver that packet in fewer than d steps. The maximum such distance a packet must traverse in a routing problem I is called the *distance* lower bound, denoted D_I. Consider now the case where s packets have their destinations inside some region of the network and these packets originate from outside the region (see Figure 1). All these packets must enter the region. If the region has z incoming links in its perimeter, then at each time step at most z packets can enter the region, and thus, no routing algorithm can deliver those s packets in fewer than s/z steps. The maximum value of this ratio, taken over all the regions in the network, for a problem instance I yields the *bandwidth* lower bound, denoted W_I. The *lower bound* for I, which we denote by LB_I, is just $\Omega(D_I + W_I)$.

A family of routing problems is *hard* if the trivial lower bound for each of its members is $\Omega(n)$. This paper introduces a new hot-potato routing algorithm that solves any hard batch routing problem I with high probability (at least $1-\frac{1}{n}$) in time $O(LB_I \cdot \log^3 n)$. This algorithm is the first hot-potato algorithm whose performance, with high probability, lies within a polylogarithmic factor of optimal for a non-trivial class of batch routing problems.

Our algorithm is *distributed*: each node makes routing decisions based on its local state, independently of the other nodes. Moreover, nodes know nothing about the initial distribution of destinations (including the values of D_I, W_I, and LB_I).

At the heart of our algorithm is a new technique based on *multi-bend* paths, a departure from the paths using a constant number of bends used in most other hot-potato algorithms. Each time a packet is deflected (unable to advance toward its destination), it may, with a certain probability, become *excited*, increasing its priority over non-excited packets. An excited packet attempts to converge on its target by choosing a logarithmic number of random intermediate destinations (see Figures 2 and 4) in a sequence of squares of decreasing size. As we will show in the analysis, a packet during its multi-bend path has a good chance not to be interrupted by other high-priority packets, and therefore, to successfully reach its destination.

We proceed as follows. In Section 2 we present related work. In Section 3 we give some necessary preliminary definitions and notations. We present our algorithm in Section 4 and in Section 5 we give its time analysis. We describe the performance of our algorithm in terms of the trivial lower bound in Section 6. We conclude in Section 7 with a discussion and open problems.

2. RELATED WORK

Hot-potato routing was first proposed by Baran [2]. For mesh-like networks, there are many hot-potato algorithms tuned for the batch permutation and random destination routing problems [7; 8; 11; 12; 15; 18]. In the permutation problem every node is the source and destination of exactly one packet, and in the random destinations problem every packet chooses its destination at random.

In the more general setting of arbitrary many-to-one batch routing problems, there are several known hot-potato algorithms. Using potential function analysis, Ben-Dor *et al.* [5] provide a simple algorithm for the 2-dimensional $n \times n$ mesh with $O(n\sqrt{k})$ steps, where k is the total number of packets to be routed. They generalized their techniques for the d-dimensional mesh to obtain $O(e^d n^{d-1} k^{1/d})$ steps. Borodin *et al.* [6] present a hot-potato routing algorithm for the d-dimensional mesh with $D_I + 2(k-1)$ steps, where D_I is the distance lower bound for any routing problem instance I. Similarly, Ben-Aroya *et al.* [3] give an algorithm that finishes in $D_I + 2(k-1)$ steps in the two-dimensional mesh. For single target problems, Ben-Aroya *et al.* [4] give a randomized algorithm for the d-dimensional mesh that finishes in $O(k/d)$ steps, with high probability.

For the problems we consider here, in which there are n^2 packets to be routed, all these algorithms require $O(n^2)$ steps, which can be achieved by a naive solution. The algorithm presented here is the first that does better.

Solving arbitrary routing problems is difficult even for the traditional store-and-forward routing algorithms. The best store-and-forward algorithm is due to Mansour and Patt-Shamir [13], and performs within a factor of $\log(D_I)$ of the lower bound LB_I for any routing problem instance I. In their algorithm the nodes have buffers of size $\log(D_I)$. There was no known similar result for hot-potato algorithms. It was surprising to find that for hard routing problems, in which D_I is $\Omega(n)$, our hot-potato algorithm matches this bound within a polylogarithmic factor even though it uses no buffers.

3. PRELIMINARIES

We are given an $n \times n$ mesh of nodes. We denote a node v with its coordinates (x, y), $0 \le x, y < n$, where x is a column and y a row. The lower-left corner of the mesh is the node $(0,0)$ and the upper-right corner $(n-1, n-1)$. Each node (except at the edge of the mesh) is connected to its four adjacent nodes by bidirectional links, denoted *up*, *down*, *left* and *right*.

We denote the *distance* between nodes $v = (x, y)$ and $v' = (x', y')$ as $dist(v, v')$ and is the quantity

$$dist(v, v') = |x - x'| + |y - y'|.$$

This distance measures how long it takes an undeflected packet to travel from v to v'. This distance is sometimes called the *Manhattan* metric or L_2 *norm*.

We denote a rectangle with lower leftmost node v and upper rightmost node v' as $[v, v']$. When necessary we distinguish between the binary logarithm lg and the natural logarithm ln.

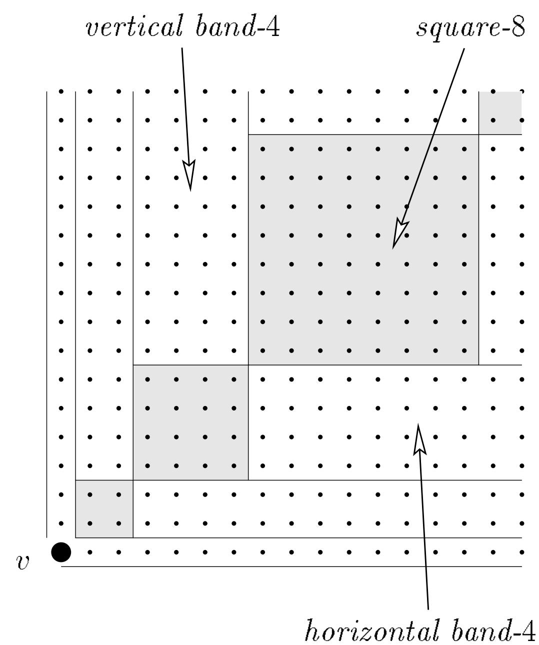

Figure 2: The squares and bands of a node

Take a node $v = (x, y)$ and a number $z = 2^k$, where $k = 0, \ldots, \lg n - 1$. Consider the sub-mesh that is up and right from v. The *square-z* of v is the $z \times z$ square whose lower leftmost node is

$$v' = (x + z - 1, y + z - 1),$$

as shown in Figure 2. If the *square-z* does not fit entirely into the mesh, it is truncated at the mesh boundary. Note that *square*-1 is the node v itself. The *horizontal band-z* of v is the rectangle

$$[(x+2z-1, y+z-1), (n-1, y+2z-1)].$$

The *vertical band-z* of v is the rectangle

$$[(x+z-1, y+2z-1), (x+2z-1, n-1)].$$

Note that all *square-z*, *horizontal band-z*, *vertical band-z*, for $z = 2^k$ and $k = 0, \ldots, \lg n - 1$, partition the sub-mesh that is up and right from v (say, the rectangle $[v, (n-1, n-1)]$). That is, every node in that sub-mesh can be assigned to exactly one *square* or *band*. By symmetry, we define the squares and bands of v in the other three sub-meshes. Similarly, we define the squares and bands for any node in the network.

4. ALGORITHM

Our algorithm is *greedy*: in a node, a packet always tries to follow any link that brings it closer to its destination. In case a packet cannot follow any such link, because other packets will occupy these links, then it is forced to follow some other link that takes it further away from its destination, and in this case we say that the packet is *deflected*. When two or more packets are competing in a node for the same link we say that there is a *conflict*.

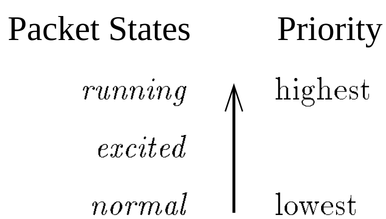

Figure 3: Packet states and priority

In our algorithm there are three states for a packet: *normal*, *excited*, *running*. A packet is in only one of these states. Each packet state corresponds to a priority and the *running* packets have the highest priority and the *normal* packets have the lowest (see Figure 3). The priorities determine how the conflicts are resolved in the nodes, namely, the higher priority packets win over the lower priority packets. Conflicts between packets of the same priority are resolved in an arbitrary way (except for the packets in the *running* state for which we describe below how the conflicts are resolved). For implementation purposes, each packet can be divided into an immutable message part, and a mutable header containing the packet's priority (two bits suffice).

Initially, all the packets are in the *normal* state. A packet in the *normal* state simply tries to follow any of the available links that brings it closer to its destination. A *normal* packet is deflected whenever other *normal* or higher priority packets have already taken all the links that could bring it closer to the destination.

Consider now a *normal* packet π which in the previous time step was deflected from some node at distance $d-1$ from its destination, so that in the current time step it appears at node v at distance d from its destination (see Figure 4). The deflected packet π becomes *excited* at node v with probability p, and otherwise, it remains *normal* with probability $1-p$. The probability p changes over time and it is given by the function

$$p(t) = \frac{c \ln t}{t},$$

where c is a constant we will specify later. To avoid notational clutter, we use p to denote $p(t)$, when t is clear from context. Similarly, any packet becomes *excited* with probability p whenever it is deflected.

Let's assume now that packet π becomes *excited* in node v, and that, without loss of generality, packet π has its destination node lower and left from v. The algorithm assigns each *excited* packet a *preferred link* that brings it closer to its destination. If the node v is in a *horizontal band-z* of packet π's destination node, the packet π prefers to take the left link first (see Figure 4). If the node v is in a *vertical band-z* or in a *square-z* of packet π's destination node, the *excited* packet π prefers to take the down link first.

The *excited* packet π will try to follow its preferred link. There are only two cases that the *excited* packet π will not be able to follow its preferred link. The first case is when the *excited* packet π conflicts in node v with other excited packets that wish to follow the same preferred link. Such a conflict is resolved in an arbitrary way and packet π may lose. The second case is when packet π conflicts with a *running* packet, which has higher priority. The *running* packets have also a preferred link. If the *excited* packet π conflicts with a *running* packet for the same preferred link then packet π always loses. In both cases, if packet π loses then it loses its high priority and it enters immediately (the same time step) the *normal* state, and it will be treated in node v as a *normal* packet.

If the *excited* packet π succeeds in taking its preferred link, then in the new node, one link closer to its destination, the packet π changes its state again and it becomes a *running* packet. (Notice that packet π stays in the *excited* state for at most one time step.) Similar to the *excited* packets, any *running* packet has a preferred link that brings it closer to its destination. As long as the *running* packet succeeds in following its preferred link it remains in the *running* state until it reaches its destination, in which case the packet π is absorbed. If in some node the *running* packet is unable to follow its preferred link, because of conflicts with other *running* packets with the same preferred link, then it enters immediately the *normal* state. The preferred links for a *running* packet are chosen as follows (see Figure 4).

- If the *running* packet π was *excited* in a node v in the *horizontal band-z* of its destination, then it will go directly to a random node (1 of z) on the same row in *square-z* by following the left links repeatedly. This is the case of Figure 4. (From *horizontal band*-1, the packet tries to go directly to its destination.)

- If the *running* packet π was *excited* in a *vertical band-z* it will go to a random node in *square-$z/2$* by following a one-bend path. The first part of a one-bend path has direction down and the second part has direction left. (From *vertical band*-1, the packet tries to go directly to its destination.)

- If the node v is in a *square-z* of packet π's destination node, it will go to a random node in *square-$z/2$* by following a one-bend path, first down then left.

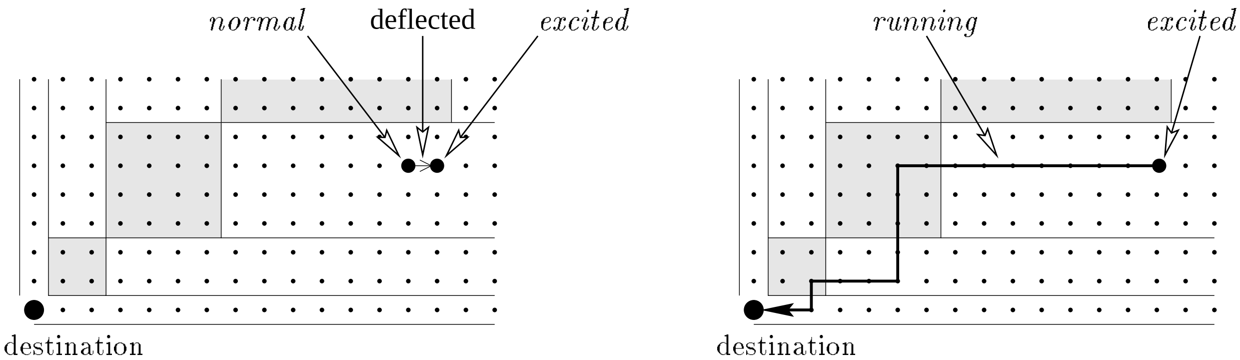

Figure 4: Left: a deflected packet becomes *excited*. Right: a successful *running* path

This procedure is repeated for squares $z/2, z/4, \ldots$ until the *running* packet reaches its destination node (that is, $z = 1$). A *running* packet *turns* whenever the link it exits a node is not opposite to the link the packet has entered the node. Note that a *running* packet can conflict in a node with another *running* packet, with the same preferred link, only when it turns. In such a case we specify that a turning packet loses to a non-turning packet. There are no conflicts between non-turning packets, and conflicts between turning packets are resolved in an arbitrary way. For convenience, let us call a *running* packet before its first turn a *runningA* packet; a *running* packet on or after its first turn is called a *runningB* packet.

5. TIME ANALYSIS

In this section we give the time analysis of our algorithm. Every randomized algorithm needs an adversary. In this paper, we assume that every time a packet π becomes *excited* and then *running*, the adversary can attempt to "launch" *normal* packets on a collision course that will conflict with π. The adversary launches a packet by deflecting it in the hope it will become *excited* and proceed to collide with π. If that packet fails to become *excited*, however, then it will remain *normal* which has lower priority than π, and cannot interrupt π. The key insight underlying this part of the analysis is that the adversary has at most one chance to launch any particular packet on a collision course with π, thus the adversary's "ammunition" is limited. Moreover, as explained below, by making the adversary so powerful, a packet's chances of traveling uninterrupted to its destination after successive excitations can be treated as independent events.

A key term in the analysis is the parameter m which for any batch problem instance has to do with the distribution of the destinations in square regions of the network.

DEFINITION 5.1. *Let S be a square subregion of the mesh, and let $dest(S)$ be the number of packets with destinations in S. The parameter m is defined as the maximum number so that no $k \times k$ square has more than mk destination packets. Explicitly,*

$$m := \max_{k=1}^{n} \frac{1}{k} \cdot \max_{S} dest(S)$$

as S ranges over all $k \times k$ square subsets of the mesh.

The parameter m is related to the bandwidth lower bound and this relationship is explored in Section 6. Although our complexity analysis depends on m, the value of m is not known to the routing algorithm.

We immediately obtain the following lemma.

LEMMA 5.2. $n \le m \le n^2$.

PROOF. There are n^2 packets to route, so the maximum possible value of $dest(S)$ is n^2. Taking S to be the entire mesh,

$$m \ge \frac{dest(S)}{n} = n.$$

Taking S to be a single node,

$$m \le n^2.$$

□

In the analysis we will make use of the following inequalities. For all n, t, such that $n \ge 1$ and $|t| \le n$,

$$e^t \left(1 - \frac{t^2}{n}\right) \le \left(1 + \frac{t}{n}\right)^n \le e^t. \qquad (1)$$

For all p, k, such that $0 < p < 1$ and $k \ge 1$,

$$1 - p \le \left(1 - \frac{p}{k}\right)^k. \qquad (2)$$

We continue in this section as follows. First we prove in Sections 5.1 and 5.2 that the probability there will be an *excited* or *running* packet at a given node and time is small. In Section 5.3 we will see that a deflected packet has a good chance to be *excited* and *running* to its destination without any interruptions. Finally, in Section 5.4 we prove that with high probability each packet will have arrived at its destination after the promised time.

5.1 The Excited State

To avoid notational clutter in this section and in Sections 5.2 and 5.3, we will assume that all events are happening after time t_0, where the exact value of t_0 will be specified in Section 5.4. Furthermore, we will assume that the probability of becoming excited at time $t_0 - 2n$ is p, and thus, after time $t_o - 2n$ a packet becomes *excited* with probability at most p (since this probability decreases with time).

LEMMA 5.3. *The probability that a particular node contains no excited packet is at least $(1 - p)^4$.*

PROOF. A deflected packet becomes *excited* with probability at most p only if it was deflected in the preceding time step. It will fail to become *excited* with probability at least $1-p$. Since at any time step a node contains at most four packets, all four will fail to become *excited* with probability at least $(1-p)^4$. $\square$

5.2 The Running State

LEMMA 5.4. *The probability that at node v and time t there is no runningA packet is at least $(1-p)^{8n-8}$.*

PROOF. Let's assume that at a node $v=(x,y)$ at time t there is a *runningA* packet π. Packet π must have become *excited* at some node v' in the same row or column at time $t'=t-dist(v,v')$, for $t-t'=1,\ldots,n-1$. From Lemma 5.3, node v' will not contain any *excited* packet at time t' with probability at least $(1-p(t'))^4 \geq (1-p)^4$. All the nodes v' (there are at most $2n-2$) will contain no *excited* packets at the corresponding time steps t' with probability at least

$$(1-p)^{4(2n-2)} = (1-p)^{8n-8}.$$

$\square$

A *runningB* packet π can always be considered as being on a one-bend path from *square-z* to *square-z/2*; we say that the packet π is a *runningB*(z) packet.

LEMMA 5.5. *A runningB(z) packet π chooses a particular row or column with probability at most $2/z$.*

PROOF. If the *runningB*(z) packet π is on the first part of the one-bend path (thus the preferred link is a column link), then it has chosen randomly one of z columns. If the *runningB*(z) packet π is on the second part of the one-bend path (thus the preferred link is a row link), then it has chosen randomly one of $z/2$ rows. $\square$

For the rest of this section, we will assume that the packets have destinations down and left.

LEMMA 5.6. *A runningB(z) packet π at node $v=(x,y)$ has its destination inside the square*

$$S := [(x-2z+2, y-2z+2), (x-\frac{z}{2}+1, y-\frac{z}{2}+1)].$$

PROOF. The current position v of packet π must be somewhere in the square $S'=[v',v'']$, with v' being the lower left corner of *square-z/2* and v'' being the upper right corner of *square-z* of packet π's destination. If v coincides with node v' then the destination has coordinates

$$(x-\frac{z}{2}+1, y-\frac{z}{2}+1).$$

If v coincides with node v'' then the destination has coordinates

$$(x-2z+2, y-2z+2).$$

Subsequently, the destination of packet π is inside the square S. $\square$

Let π be a packet in the *runningB*(z) state. We say that π *might arrive* at node v if there is some execution in which π arrives at v in the *runningB*(z) state.

LEMMA 5.7. *The number of runningB(z) packets that might arrive at node v is at most $3/2 \cdot mz$.*

PROOF. With Lemma 5.6 only packets with destination inside square S can be *runningB*(z) packets at node v. The size of this square is smaller than

$$\frac{3z}{2} \times \frac{3z}{2}.$$

By Definition 5.1 the square S cannot have more than $m \cdot 3z/2$ destination packets. $\square$

LEMMA 5.8. *A node v contains no runningB(z) packet at time t with probability at least*

$$\left(1-\frac{2p}{z}\right)^{\frac{3}{2}\cdot mz}.$$

PROOF. Consider a *runningB*(z) packet π that is at a node v at time t. Packet π has at most one chance to get *excited* and appear in v at time t. This chance is given to packet π at some time $t'<t$ and at a node v' with distance $t-t'$ from v. Packet π must have been deflected at time $t'-1$ and at time t' in node v' it becomes *excited* and follows a *running* path to node v. If packet π loses this chance and becomes *excited* in a subsequent deflection then it will fail to arrive at node v at time t and it will arrive there some time after t, since any subsequent deflection takes packet π further from node v by a link.
The probability of π getting *excited* at v' is at most $p(t')<p$. If the *runningB*(z) packet π enters node v when it is in the column part of its path from *square-z* to *square-z/2*, then from Lemma 5.5 it chooses the column of node v with probability at most $2/z$, and similarly for the row part. Subsequently, the probability that packet π appears in v at time t is at most $2p/z$ and the probability that it doesn't appear in v at all is at least

$$1-\frac{2p}{z}.$$

According to Lemma 5.7, the number of possible packets like π is at most $3/2 \cdot mz$. Therefore, none of them will be in node v and time t with probability at least

$$\left(1-\frac{2p}{z}\right)^{\frac{3}{2}\cdot mz},$$

as needed. $\square$

LEMMA 5.9. *The probability that at a node v there is no runningB packet at time t is at least $(1-p)^{3m(\lg n-1)}$.*

PROOF. A *runningB* packet can be a *runningB*(z) packet, with $z=2^k$ and k is one of $1,\ldots,\lg n-1$. Considering all the values of z, by Lemma 5.8 and applying Equation 2, we have that there will be no *runningB* packet at node v and time t with probability at least

$$\begin{aligned}
\prod_{k=1}^{\lg n-1}\left(1-\frac{2p}{2^k}\right)^{\frac{3}{2}\cdot m\cdot 2^k} &= \prod_{k=1}^{\lg n-1}\left(1-\frac{p}{2^{k-1}}\right)^{3m\cdot 2^{k-1}} \\
&\geq \prod_{k=1}^{\lg n-1}(1-p)^{3m} \\
&= (1-p)^{3m(\lg n-1)}.
\end{aligned}$$

$\square$

5.3 One Packet

LEMMA 5.10. *At a node v at time t there is no excited or running packet with probability at least $(1-p)^{12m\lg n}$.*

PROOF. The probability that no *excited* or *running* (*runningA* or *runningB*) packet is at node v and time t is the product of the probabilities that each of them is not at node v and time t.

- By Lemma 5.3, there is no *excited* packet at node v and time t with probability at least $(1-p)^4$.
- By Lemma 5.4, there is no *runningA* packet at node v and time t with probability at least $(1-p)^{8n-8}$.
- By Lemma 5.9, there is no *runningB* packet with destination down and left at node v and time t with probability at least $(1-p)^{3m(\lg n-1)}$. Since there are four symmetric cases, the probability to have no *runningB* packet at all is at least $(1-p)^{12m(\lg n-1)}$.

From Lemma 5.2 we know that $m \geq n$. Therefore we can bound the product (for the sake of simplicity)

$$(1-p)^4 \cdot (1-p)^{8n-8} \cdot (1-p)^{12m(\lg n-1)} > (1-p)^{12m\lg n}.$$

□

We say that an *excited* packet travels *uninterrupted* to its destination if it becomes *running* and reaches its destination without encountering any conflicts.

THEOREM 5.11. *When a packet becomes excited it will travel uninterrupted to its destination with probability at least $(1-p)^{24m\lg^2 n}$.*

PROOF. By Lemma 5.10 there is no other *excited* or *running* packet at any specific node v and time t with probability at least

$$q := (1-p)^{12m\lg n}.$$

After a packet π becomes *excited* it will become *running* and it will turn at most $2(\lg n - 1)$ times before arriving at the destination. Each of those turns, and the single transition from *excited* to *running*, will be successful with probability at least q, since with at least this probability it will not conflict with other packets. Whenever the *running* packet π is not turning, it will successfully take its preferred link since it is the only packet entering from the opposite link. Therefore, an *excited* packet will succeed to reach its destination with probability at least

$$q \cdot q^{2(\lg n-1)} \geq (1-p)^{24m\lg^2 n}.$$

□

5.4 All Packets

Our algorithm satisfies some interesting properties with high probability (at least $1-1/n$). We use the following constants and time steps.

$$c = 24e \qquad c' = 3\cdot 24c\lg^2 e$$
$$t_0 = c'm\ln^3 n + 2n \qquad t_1 = 3c'm\ln^3 n$$

Recall that the probability of becoming excited is

$$p(t) = \frac{c\ln t}{t}.$$

If the value of m were known to the algorithm, then p would not need to vary with time.

LEMMA 5.12. *If packet π becomes excited at time $t \geq t_0$, then the probability of reaching its destination is at least $1/2e$.*

PROOF. Any packet conflicting with packet π must have been *excited* at most $2n$ steps before π. The probability that such a packet became *excited* was at most

$$p = p(t_0 - 2n) = \frac{c\ln(c'm\ln^3 n)}{c'm\ln^3 n}.$$

By Theorem 5.11, packet π will reach its destination with probability at least $(1-p)^{24m\lg^2 n}$. Since $m \leq n^2$ (Lemma 5.2), by taking n to be sufficiently large, such that $c'\ln^3 n \leq n$ (thus $c'm\ln^3 n \leq n^3$), and by Equation 1 we get

$$\begin{aligned}
(1-p)^{24m\lg^2 n} &= \left(1 - \frac{c\ln(c'm\ln^3 n)}{c'm\ln^3 n}\right)^{24m\lg^2 n} \\
&\geq \left(1 - \frac{3c\ln n}{c'm\ln^3 n}\right)^{24m\lg^2 n} \\
&= \left(1 - \frac{1}{24m\lg^2 n}\right)^{24m\lg^2 n} \\
&\geq \frac{1}{e}\left(1 - \frac{1}{24m\lg^2 n}\right) \\
&\geq \frac{1}{2e}.
\end{aligned}$$

□

LEMMA 5.13. *Each time t (with $t_0 \leq t < t_1$) a packet is deflected, it will arrive uninterrupted at its destination with probability at least*

$$\frac{c}{6ec'm\ln^2 n}.$$

PROOF. Whenever a packet is deflected with probability at least $p(t_1)$ it becomes *excited*. Since $n \leq m$ (Lemma 5.2), we get

$$\begin{aligned}
p(t_1) &= \frac{c\ln(3c'm\ln^3 n)}{3c'm\ln^3 n} \\
&> \frac{c\ln n}{3c'm\ln^3 n} \\
&= \frac{c}{3c'm\ln^2 n}.
\end{aligned}$$

The probability of a packet arriving at its destination without interruptions when becoming *excited* is according to Lemma 5.12 at least $1/2e$. Therefore, the probability of the packet arriving at its destination without interruptions when being deflected is at least

$$\begin{aligned}
p(t_1)\cdot\frac{1}{2e} &= \frac{c}{3c'm\ln^2 n}\cdot\frac{1}{2e} \\
&= \frac{c}{6ec'm\ln^2 n}.
\end{aligned}$$

□

LEMMA 5.14. *If a packet π is deflected x times, then it will reach its destination in at most $2x + 2n - 2$ steps.*

PROOF. Initially, the distance of π's source to its destination is no more than $2n-2$. Each time π is deflected the distance increases by a link, and each time it follows a link closer to its destination the distance decreases. □

LEMMA 5.15. *With probability at least $1-1/n^3$, a packet will reach its destination in t_1 steps.*

PROOF. By Lemma 5.13, each time that a packet π is deflected in the time interval $t_0 \le t < t_1$, packet π will arrive at its destination without interruptions with probability at least

$$q := \frac{c}{6ec'm\ln^2 n}.$$

Because the adversary is allowed to redistribute the other packets at each deflection, successive probabilities are independent. By Lemma 5.14, the number of deflections of packet π that can fit in the time interval $t_0 \le t < t_1$ is at least

$$\begin{aligned} x &:= \frac{(t_1 - t_0 - 2n)}{2} \\ &= \frac{(3c'-c')m\ln^3 n - 4n}{2} \\ &= c'm\ln^3 n - 2n. \end{aligned}$$

Since $n \le m$ (Lemma 5.2), we obtain

$$\begin{aligned} x &= c'm\ln^3 n - 2n \\ &\ge c'm\ln^3 n - \frac{1}{4}c'm\ln^3 n \\ &= \frac{3}{4}c'm\ln^3 n. \end{aligned}$$

Packet π will fail to reach its destination after x deflections with probability at most $(1-q)^x$. By Equation 1, we get

$$\begin{aligned} (1-q)^x &\le \left(1 - \frac{c}{6ec'm\ln^2 n}\right)^{\frac{3}{4}c'm\ln^3 n} \\ &= \left(1 - \frac{3\ln n}{\frac{3}{4}c'm\ln^3 n}\right)^{\frac{3}{4}c'm\ln^3 n} \\ &\le e^{-3\ln n} \\ &= \frac{1}{n^3}. \end{aligned}$$

□

THEOREM 5.16. *For any batch problem instance, with high probability (at least $1-1/n$), all packets reach their destination nodes in at most $O(m\ln^3 n)$ steps.*

PROOF. By Lemma 5.15, a packet will arrive at its destination in t_1 steps with probability at least $1-1/n^3$.
Since we have made a worst case analysis for each packet by assuming that the adversary can reorganize all other packets whenever one is deflected, we can safely assume that the packets are independent of each other in the analysis.
Therefore, the probability that all n^2 packets will arrive at their destinations within t_1 steps is at least (applying Equation 2)

$$\left(1 - \frac{1}{n^3}\right)^{n^2} = \left(1 - \frac{1/n}{n^2}\right)^{n^2} \ge 1 - \frac{1}{n}.$$

□

6. LOWER BOUND

In this section, we show how our algorithm relates to the trivial lower bound

$$LB_I = \Omega(D_I + W_I).$$

Recall that for any batch problem instance I the bandwidth lower bound W_I for a region S is defined by taking the number of packets s_1 with origins outside S and destinations inside S, divided by the *perimeter* of S, the number of links leading into S.

DEFINITION 6.1. *For any instance I of a routing problem, and any region S of the mesh, let $dest(S)$ be the number of packets in I with destinations in S (independently of their origins), $perim(S)$ the perimeter of S, and $M_I(S) = dest(S)/perim(S)$.*

$$M_I = \max(M_I(S)) \textit{ where } S \textit{ ranges over all regions } S.$$

It is immediate that $m/4 \le M_I$, since $m/4$ is the maximum M_I taken over square regions only (see Definition 5.1).
For any region S, let s_0 be the number of packets in I with sources within S, and s_1 be the number of packets with sources outside S.

$$\begin{aligned} M_I(S) &= \frac{s_0 + s_1}{perim(S)} \\ &= \frac{s_0}{perim(S)} + \frac{s_1}{perim(S)} \\ &= \frac{s_0}{perim(S)} + W_I(S) \\ &\le n + W_I(S). \end{aligned}$$

Subsequently,

$$M_I \le n + W_I.$$

It follows that $m/4$ differs from W_I by an additive term of at most n.
If the problem I is hard then LB_I is $\Omega(n)$, so $W_I = \Omega(n)$ or $D_I = \Omega(n)$. If $W_I = \Omega(n)$, then

$$\Omega(M_I) \le \Omega(n + W_I) = \Omega(W_I) \le \Omega(LB_I).$$

If $D_I = \Omega(n)$, then

$$\Omega(M_I) \le \Omega(n + W_I) = \Omega(D_I + W_I) = \Omega(LB_I).$$

In either case,

$$\Omega(m) \le \Omega(M_I) \le \Omega(LB_I).$$

Substituting in Theorem 5.16, we obtain our main result.

COROLLARY 6.2. *For any instance I of a hard batch problem, with high probability, all packets reach their destination nodes in at most $O(LB_I \cdot \ln^3 n)$ steps.*

7. DISCUSSION

So far we have considered only the $n \times n$ mesh. In the $n \times n$ *torus*, each node $(i, n-1)$ has a link to node $(i, 0)$, and each node $(n-1, i)$ has a link to node $(0, i)$. Our algorithm carries over to the torus with a slight (constant-factor) improvement in the time bounds, because all worst-case distances are shorter.
This algorithm raises a number of open problems. The most obvious concerns "easy" batch problems with sublinear lower bounds. We do not know whether the class of easy routing problems would yield to the same (or similar) algorithm with a more refined complexity analysis, or whether a different algorithm is needed. It would also be

interesting to consider whether this kind of multi-bend algorithm could be adapted to networks of dimension higher than two, or to networks with a different topological structures. (Note that the distance and bandwidth lower bounds apply to arbitrary networks.)

8. REFERENCES

[1] A. S. Acampora and S. I. A. Shah. Multihop lightwave networks: a comparison of store-and-forward and hot-potato routing. In *Proc. IEEE INFOCOM*, pages 10–19, 1991.

[2] P. Baran. On distributed communications networks. *IEEE Transactions on Communications*, pages 1–9, 1964.

[3] I. Ben-Aroya, T. Eilam, and A. Schuster. Greedy hot-potato routing on the two-dimensional mesh. *Distributed Computing*, 9(1):3–19, 1995.

[4] I. Ben-Aroya, I. Newman, and A. Schuster. Randomized single-target hot-potato routing. *Journal of Algorithms*, 23(1):101–120, Apr. 1997.

[5] A. Ben-Dor, S. Halevi, and A. Schuster. Potential function analysis of greedy hot-potato routing. *Theory of Computing Systems*, 31(1):41–61, Jan./Feb. 1998.

[6] A. Borodin, Y. Rabani, and B. Schieber. Deterministic many-to-many hot potato routing. *IEEE Transactions on Parallel and Distributed Systems*, 8(6):587–596, June 1997.

[7] C. Busch, M. Herlihy, and R. Wattenhofer. Randomized greedy hot-potato routing. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 458–466, Jan. 2000.

[8] U. Feige and P. Raghavan. Exact analysis of hot-potato routing. In IEEE, editor, *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science*, pages 553–562, Pittsburgh, PN, Oct. 1992. IEEE Computer Society Press.

[9] A. G. Greenberg and J. Goodman. Sharp approximate models of deflection routing. *IEEE Transactions on Communications*, 41(1):210–223, Jan. 1993.

[10] W. D. Hillis. *The Connection Machine*. MIT press, 1985.

[11] C. Kaklamanis, D. Krizanc, and S. Rao. Hot-potato routing on processor arrays. In *Proceedings of the 5th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 273–282, Velen, Germany, June 30–July 2, 1993. SIGACT and SIGARCH.

[12] M. Kaufmann, H. Lauer, and H. Schroder. Fast deterministic hot-potato routing on meshes. In Springer-Verlag, editor, *Proc. of the 5th International Symposium on Algorithms and Computation (ISAAC), Lecture Notes in Computer Science*, volume 834, pages 333–341, 1994.

[13] Y. Mansour and B. Patt-Shamir. Many-to-one packet routing on grids. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on the Theory of Computing*, pages 258–267, 29 May–1 June 1995.

[14] N. F. Maxemchuk. Comparison of deflection and store and forward techniuqes in the Manhattan street and shuffle exchange networks. In *Proc. IEEE INFOCOM*, pages 800–809, 1989.

[15] I. Newman and A. Schuster. Hot-potato algorithms for permutation routing. *IEEE Transactions on Parallel and Distributed Systems*, 6(11):1168–1176, Nov. 1995.

[16] C. L. Seitz. The caltech mosaic C: An experimental, fine-grain multicomputer. In *4th symp. on Parallel Algorithms and Architectures*, June 1992. Keynote Speech.

[17] B. Smith. Architecture and applications of the HEP multiprocessor computer system. In *Proc. Fourth Symp. Real Time Signal Processing IV*, pages 241–248. SPIE, 1981.

[18] P. Spirakis and V. Triantafillou. Pure greedy hot-potato routing in the 2-D mesh with random destinations. *Parallel Processing Letters*, 7(3):249–258, Sept. 1997.

[19] T. Szymanski. An analysis of "hot potato" routing in a fiber optic packet switched hypercube. In *Proc. IEEE INFOCOM*, pages 918–925, 1990.

[20] Z. Zhang and A. S. Acampora. Performance analysis of multihop lightwave networks with hot potato routing and distance age priorities. In *Proc. IEEE INFOCOM*, pages 1012–1021, 1991.