Randomized Greedy Hot-Potato Routing

Costas Busch, Maurice Herlihy, and Roger Wattenhofer
Computer Science Department,
Brown University,
Providence, RI 02912
{cb,herlihy,roger}@cs.brown.edu [*]

Abstract

We present a novel greedy hot-potato routing algorithm for the 2-dimensional $n \times n$ mesh or torus. This algorithm uses randomization to adjust packet priorities. For any permutation problem or random destination problem, it ensures that each packet reaches its destination in asymptotically optimal expected $O(n)$ steps, and all packets reach their destinations in $O(n \ln n)$ steps with high probability, an improvement over the previously-known deterministic upper bound of $O(n^2)$ for greedy algorithms. For a general batch problem, with high probability all packets reach their destination nodes in at most $O(m \ln n)$ steps, where $m = \min(m_r, m_c)$, where m_r and m_c are respectively the maximum number of packets targeted to a single row or column.

1 Introduction

A *hot-potato* (or *deflection*) routing algorithm is a packet routing algorithm in which nodes are unable to buffer 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 are interesting because they have been observed to work well in practice: hot-potato routing algorithms have been used in parallel machines such as the HEP multiprocessor [20], the Connection machine [12], and the Caltech Mosaic C [19], as well as high speed communication networks [16]. Hot-potato routing algorithms are well-suited for optical networks [1, 10, 16, 22, 23] because it is difficult to buffer optical messages.

A hot-potato routing algorithm is *greedy* [3, 6] if each node forwards each packet closer to its destination whenever possible (that is, whenever the desired links are not already assigned to other advancing packets). A packet that does not advance toward its destination is said to be *deflected.* Greedy algorithms are particularly attractive because they tend to be simple, admitting efficient hardware implementations. Greedy algorithms are also adaptive — when contention is low, packets follow the shortest routes to their destinations. Because greedy algorithms are local, they are well-suited to problems in which packets are injected dynamically.

This paper presents a new greedy hot-potato routing algorithm for the 2-dimensional $n \times n$ mesh or torus. Like some earlier algorithms [6, 11], we assign priorities to packets. Each packet is divided into an immutable message part, and a mutable header containing the packet's priority (three bits suffice). A novel aspect of our algorithm is the way it exploits randomization to adjust priorities. Each time a packet is deflected, there is a small probability it will attempt a *home run*: it increases its priority and attempts to travel by a one-bend path directly to its destination[1].

Ben-Dor *et al.* [6, Section 1.1] remark:

> Although fairly simple greedy hot-potato algorithms perform very well in practice, they resist formal analysis attacks.

The paper makes the following contributions. Both the home run technique and its analysis are novel. The algorithm is simple, relying on a fixed number of priorities. For any permutation problem or random destination problem, it ensures that each packet reaches its destination in asymptotically optimal expected $O(n)$ steps, and all packets reach their destinations in $O(n \ln n)$ steps with high probability, an improvement over the previously-known deterministic upper bound of $O(n^2)$ for greedy algorithms [6].

An important characteristic of a general batch problem is the maximum number of packets addressed to any single row or column. More precisely,

DEFINITION 1.1. *Given a routing problem, let m_r and m_c be respectively the maximum number of packets*

[*]The first and second authors were supported by NSF grant DMS-9505949, and the third by the Swiss National Science Foundation.

[1]The American author insists on pointing out that to baseball purists, this action may appear closer to stealing a base than to hitting a home run.

targeted to a single row or column. Define

$$m := \max(n, \min(m_r, m_c)).$$

Many of our complexity results are expressed in terms of m (and n). The value m is a rough reflection of a problem's inherent difficulty: high values of m imply high potential levels of congestion, and vice-versa.

Our algorithm guarantees that all packets reach their destination nodes in at most $O(m \ln n)$ steps, with high probability, even when nodes do not know the value of m. Our algorithm applies to both the 2-dimensional mesh and torus.

2 Related Work

Hot-potato routing was first proposed by Baran [3]. There are several results for the mesh and torus networks for greedy and non-greedy hot-potato routing algorithms. (For a more complete review of hot-potato algorithms, see [18].)

For greedy hot-potato routing, Ben-Dor *et al.* [6] give a potential function analysis and they 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.* [7] present a complicated deterministic greedy hot-potato routing algorithm for the d-dimensional mesh and the 2-dimensional torus where any packet p finishes in at most $\text{dist}(p) + 2(k-1)$ steps, where $\text{dist}(p)$ is the initial distance of p from its destination (they also present a simple non-greedy algorithm with similar results). For the 2-dimensional mesh and torus this algorithm preserves the $O(n^{1.5})$ bound given by Bar-Noy *et al.* [2]. For the 2-dimensional case a similar result was independently obtained by Ben-Aroya *et al.* [4]. For a single destination or a small set of destinations Ben-Aroya *et al.* [5] present a randomized algorithm on the d-dimensional mesh that finishes in $O(k/d)$ steps, with high probability.

For non-greedy hot-potato routing, Feige and Raghavan [9] present an algorithm for the $n \times n$ torus that routes any random destination problem in $2n + O(\ln n)$ steps with high probability. They also give an alternative algorithm that routes any permutation problem in $9n$ steps with high probability. Newman and Schuster [17] give a deterministic algorithm for permutation routing on the $n \times n$ mesh that finishes in $7n + o(n)$ steps and is based on sorting. This result was improved by Kaufmann *et al.* [14] to $3.5n + o(n)$ steps. Kaklamanis *et al.* [13] present an algorithm that routes most of the permutations in the d-dimensional torus in $dn/2 + O(\ln^2 n)$ steps and in the 2-dimensional mesh in $2n + O(\ln^2 n)$ steps. Bar-Noy *et al.* [2] present a simple deterministic algorithm for the $n \times n$ mesh and torus that routes any permutation problem in $O(n^{1.5})$ steps. Specifically, their algorithm routes any batch problem in $O(n\sqrt{m_c})$ steps where m_c is the maximum number of packets destined to any column. They also give a more complicated algorithm that runs in $O(n^{1+\epsilon})$ steps for every constant $\epsilon \geq 0$. Spirakis and Triantafillou [21] describe a routing algorithm for the random destination problem on the two-dimensional mesh. In this algorithm, the average time to deliver all packets, where the average is taken over all possible destination assignments, is $O(n \log n)$. Broder and Upfal [8] give a dynamic analysis of a non-greedy algorithm for the torus.

3 Preliminaries

We consider an $n \times n$ rectangular mesh of nodes, for $n > 2$. Each node has coordinates (x, y), for $0 \leq x, y < n$, where x is a column and y a row. The lower-left corner has coordinates $(0, 0)$ and the upper-right corner $(n-1, n-1)$. Each node (except at the edge of the mesh) is connected to its neighbors by four bidirectional *links*, denoted *up*, *down*, *left* and *right*.

Nodes take steps synchronously. At each step, a node receives at most one packet on each incoming link, routes them, and sends at most one packet on each outgoing link. The *distance* between nodes (x_0, y_0) and (x_1, y_1) is the quantity

$$|x_0 - x_1| + |y_0 - y_1|.$$

This distance measures how long it takes an undeflected packet to travel from (x_0, y_0) to (x_1, y_1). This distance is sometimes called the *Manhattan* metric or L_2 *norm*.

Our algorithm also applies to the $n \times n$ *torus*, in which each node with row (or column) coordinate $n-1$ has a link to the node with row (or column) coordinate 0. For brevity, we will focus here on the mesh, postponing discussion about the torus to the discussion section below.

Every packet has a destination node. A packet is *restricted* if it is on the same row or column as its destination. A *good link* for a packet is one that brings it closer to its destination, and a *bad link* is one that does not. A packet is *deflected* if it is forwarded along a bad link. A node that receives a packet can tell whether it was just deflected by comparing its destination and its link. Good *row links* and *column links* are defined in the obvious way.

The algorithm introduced here is *greedy*: a packet fails to follow a good link only if some other packet already occupies that link. Our algorithm makes use of *priorities*: each node routes higher-priority packets before routing lower-priority packets. As a result, a packet is deflected only if its good links are already taken by packets of greater or equal priority.

In a *batch* problem, each node sends at most one packet at time zero. In the *batch permutation* problem, each node sends a packet, and no two packets have the same destination. In the *random destination* problem, each node sends a packet and the destinations are chosen uniformly at random. For these problems, we are interested in the number of steps needed to deliver every packet.

4 The Algorithm

We start with a highly simplified, informal overview of our algorithm, intended to convey the intuition underlying the technical details. Initially, all packets are routed greedily with equal (low) priority. Each time a packet is deflected, however, there is a small probability that it will become *excited*, causing its priority to jump higher. When a packet becomes *excited*, it tries to take one of the two shortest "one-bend" paths to its destination, a strategy we call a *home run*. The home run succeeds if the packet arrives at its destination without further deflection.

As described below, a packet assumes several different priorities at different stages of a home run. For now, it is enough to say that a packet attempting a home run will be deflected only if it encounters certain other packets attempting "conflicting" home runs. An essential aspect of our algorithm is that any packet that attempts a home run will succeed with constant probability (that is, probability independent of n). We exploit this property to analyze both the expected and "with high probability" behavior of the algorithm.

A randomized algorithm is often best understood as a game against an adversary who attempts to frustrate the algorithm's goals. In our model, an adversary can disrupt a home run by packet π only by "launching" other excited packets on a collision course 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 have priority lower than π, and cannot deflect π. The key insight underlying this part of the analysis is that the adversary has at most one chance to launch any particular packet against π. Suppose the adversary deflects a packet σ with the intent of disrupting a home run by π. If the deflection fails to excite σ, then no subsequent action by the adversary can cause σ to disrupt π's current home run. Our analysis exploits this observation by bounding the adversary's "ammunition" against any particular attempted home run.

The algorithm assigns every packet one or more *preferred* links, and tries to forward each packet along a preferred link. A preferred link is always a good link (taking the packet closer to its destination), but not every good link is preferred. A packet can occupy the following states, where each state corresponds to a priority (see Figure 1).

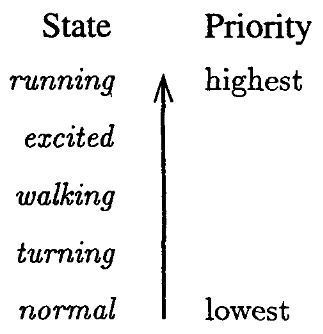

Figure 1: States and Priority

- *normal*: A packet starts out in the *normal* state. In this state, the good links are preferred: each node forwards a normal packet along one of its good links, unless those links are already occupied by other packets.

- *excited*: When a packet is deflected, it enters the *excited* state with probability p (given below), and remains *normal* with probability $(1-p)$. When a packet becomes *excited*, it tries to follow a one-bend path to its destination (a home run). It flips a coin to choose whether to traverse the row or the column first, and this choice determines its preferred link.

- *running*: If a packet starting a home run chooses to traverse the column first, it is in the $running_c$ state on the column part of its one-bend path. In this state, the good column link is preferred. If it chooses the row first, the packet is in the $running_r$ state on the row, and the good row link is preferred.

- *turning*: If a packet starting a home run chooses to traverse the column first, it enters the $turning_r$ state when it reaches its destination row, and its preferred link is along the row. If it chooses the row first, it enters the $turning_c$ state, and its preferred link is along the column.

- *walking*: If a packet starting a home run chooses to traverse the column first, it is in the $walking_r$ state as it traverses the destination row, and the preferred link is the good row link. If it chooses the row, it is in the $walking_c$ state as it traverses the column, and the preferred link is the good column link.

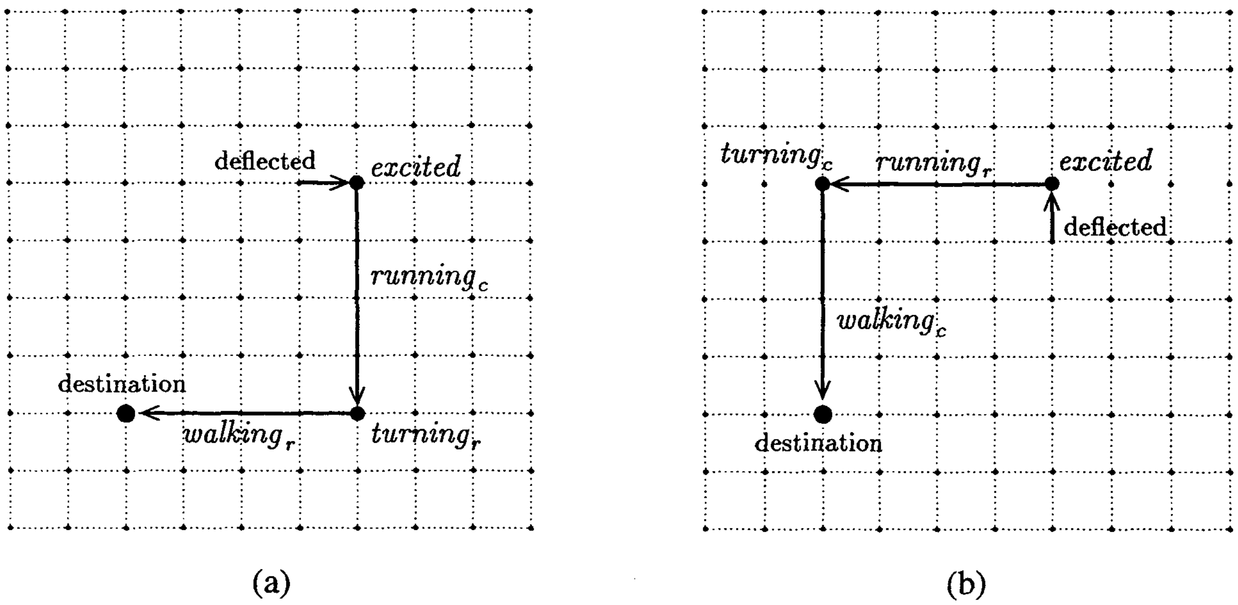

Figure 2: A Home Run: column-first and row-first traversals

If, at any point, the packet is unable to follow its one-bend path, it re-enters the *normal* state, and the node attempts to forward it along a good link. If a restricted packet becomes *excited*, it enters the *walking*$_r$ or *walking*$_c$ state directly. Figure 2(a) shows the progression of states for a packet that chooses to traverse the column first, and Figure 2(b) the row first.

If an excited packet were always to choose to traverse its row first, then the problem complexity would be determined by m_c. By choosing its row or column at random, the complexity is determined by $\min(m_c, m_r)$, performing better for problems where rows are congested but columns are not, and vice versa.

We use *running* to denote either the *running*$_c$ or *running*$_r$ states when the distinction is unimportant, and similarly for the other dual states.

At each step, a node greedily routes up to four arriving packets. Each deflected incoming packet is marked *excited* with probability p. The node then routes each of the packets in priority order. If the packet can take its preferred link, it is forwarded along that link. Otherwise the packet's priority is either reduced to *normal*, or if it is already *normal*, it is forwarded along any unoccupied link.

Two packets with the same preferred link are said to *conflict*. Conflicts between packets in the same states are resolved arbitrarily. Note that some states never conflict: for example, a packet in the *running*$_r$ state will never conflict with a packet in the *running*$_c$ state. This algorithm is greedy because a packet fails to follow any of its good links only if other packets are traversing these links.

Our algorithm is parameterized by p, the probability that a deflected packet becomes *excited*. Different values of p yield different behaviors. In the next section, we analyze the behavior of a single packet leaving p unbound. In Section 6, we compute the expected time needed for a single packet to reach its destination. With a proper choice of p, we can achieve an $O(n)$ expected time. In Section 7, we turn our attention to the time needed to solve a batch problem. With a proper (time-dependent) choice of p, we can achieve $O(m \ln n)$ time with high probability.

5 Single-Packet Analysis

We now analyze the probability that a packet π will complete a home run. When analyzing the behavior of π, we make use of the following *adversary*. Each time π becomes *excited*, the adversary is allowed to place the other packets at nodes in the mesh, and to choose their destinations, subject only to the constraint that no more than m_r or m_c packets can have destinations in the same row or column as π. The probability that an *excited* packet will complete a home run against such an adversary is a lower bound for that probability in any actual execution.

Without loss of generality, assume the packet's destination is down and left from the packet's current position. Also without loss of generality, assume the packet chooses the *running*$_c$ state (the *running*$_r$ state is symmetric). The case where the packet is restricted (already on its destination row or column) is considered below as a special case.

Recall that a deflected packet becomes excited with probability p. We will derive lower bounds on the probability for "good things" happening in terms of p, and then use these bounds to motivate our choice for p.

LEMMA 5.1. *The probability that a particular node contains no excited packet is at least $p' := (1-p)^4$.*

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

LEMMA 5.2. *An excited packet follows its preferred link and enters the running state with probability at least $(1-p)^{4n}$.*

Proof. Consider node (x, y) at time t. Our priority assignment guarantees that an *excited* packet can be deflected only by another *excited* packet, or by a *running* packet.

- By Lemma 5.1, the probability that node (x, y) has no *excited* packet at time t is at least p'.
- Recall that we assume (WLOG) the packet's preferred link is in the down direction.. A conflicting packet in the $running_c$ state must have become *excited* at node $(x, y+d)$ at time $t-d$ for it to be in the $running_c$ state at node (x, y) at time t, for $1 \leq d \leq n-y-1$. By Lemma 5.1, the probability that no packet became *excited* at those nodes and times is at least p'^{n-1}.

In total, the probability the packet will enter the *running* state is at least $p' \cdot p'^{n-1} = (1-p)^{4n}$.

LEMMA 5.3. *A packet in the running state will always proceed along its preferred link.*

Proof. Our priority rules ensure that a packet in the *running* state can be deflected only by another packet in the *running* state. Any *running* packet whose preferred link is (say) down, must have arrived from the up link. At most one packet could have arrived on the up link, so at most one *running* packet prefers to exit on the down link, and no deflection can occur.

LEMMA 5.4. *A packet π in the turning state successfully enters the walking state with probability at least $(1-p)^{4n+m_r}$.*

Proof. Consider node (x, y) at time t. Assume (WLOG) the packet arrived from the up link, and prefers the left link. Our priority rules ensure that this packet can be deflected only by an *excited* packet, a $running_r$ packet arriving from the right link, a $walking_r$ packet arriving from the right link, or by a $turning_r$ packet arriving from the down link.

- By Lemma 5.1, there is no *excited* packet at node (x, y) at time t with probability at least p'.
- A conflicting packet in the $running_r$ state must have become *excited* at node $(x+d, y)$ at time $t-d$ for it to be in the $running_r$ state at node (x, y) and time t, for $1 \leq d \leq n-x-1$. As in the proof of Lemma 5.2 the adversary will fail to excite all such packets with probability at least p'^{n-1}.
- Suppose packet σ conflicts with π at time t while σ is in the $turning_r$ or $walking_r$ states. Packet σ must have become excited at some time $t-d$, and traversed d links to collide with π, all without being deflected. Packet σ was deflected at time $t-d$ at distance d from π. The key observation is that if σ failed to become excited at that step, then it will never catch up to π in time to deflect it. From the adversary's point of view, σ had only one chance to become excited in a way that can threaten π. If it fails to become excited, then σ is no longer a threat to π. Because π is already on its destination row, there are at most $m_r - 1$ potentially conflicting packets. Therefore, with probability at least $(1-p)^{m_r}$, there is no conflicting packet in either the $turning_r$ or $walking_r$ state at node (x, y) at time t.

In total, the probability a *turning* packet will successfully become *walking* is at least $p' \cdot p'^{n-1} \cdot (1-p)^{m_r} = (1-p)^{4n+m_r}$.

The symmetric lemma holds for m_c.

LEMMA 5.5. *A packet in the walking state arrives at its destination with probability at least $(1-p)^{4n}$.*

Proof. Consider node (x, y) at time t. Assume (WLOG) that the packet's preferred link is the left link. Our priority rules imply that the packet can be deflected only by a packet in the *walking*, *running*, or *excited* states.

- The *walking* packet arrives from the right link and prefers to exit on the left. A conflicting packet in the *running* or *walking* state must also have come from the right link, which cannot happen. Therefore, the packet will not conflict with another *running* or *walking* packet.

- By Lemma 5.2, there is no *excited* packet at node (x, y) at time t with probability at least p'.

The packet will remain in the *walking* state for at most $n-2$ nodes before reaching its destination node. Therefore it will complete a home run with probability at least $p'^{n-2} > (1-p)^{4n}$.

LEMMA 5.6. *If the excited packet π chooses to follow its good column link, it will complete a home run to its destination with probability at least $(1-p)^{12n+m_r}$.*

Proof. The result follows from combining Lemmas 5.2, 5.3, 5.4, and 5.5:

$$(1-p)^{4n} \cdot 1 \cdot (1-p)^{4n+m_r} \cdot (1-p)^{4n} = (1-p)^{12n+m_r}.$$

The symmetric lemma holds for m_c.

If a packet is restricted (already in its destination row or column) when it becomes *excited*, it immediately tries to enter the *walking* state. The proof of the next Lemma follows the proof of Lemma 5.2.

LEMMA 5.7. *If a restricted packet becomes excited, it succeeds in following the good column (or row) link and entering the walking state with probability at least $(1-p)^{4n}$.*

THEOREM 5.1. *An excited packet will complete its home run with probability at least $\frac{1}{2}(1-p)^{12n+m}$.*

Proof. For unrestricted packets, the proof follows directly from Lemma 5.6, and the fact that the packet chooses each alternative (row-first or column-first) with probability 1/2. For restricted packets, it follows from Lemmas 5.5 and 5.7.

6 Expected Case Analysis

We now analyze the expected number of steps for a specific packet π to arrive at its destination. For now, we assume that the expected number of packets with the same destination row or column is n. This assumption applies, for example, to the batch permutation and random destination problems.

We make frequent use of the following inequalities. For all n, t, such that $n \geq 1$ and $|t| \leq n$,

$$e^t\left(1-\frac{t^2}{n}\right) \leq \left(1+\frac{t}{n}\right)^n \leq e^t. \tag{6.1}$$

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

$$1-p \leq \left(1-\frac{p}{k}\right)^k. \tag{6.2}$$

We now consider the behavior of the algorithm when p, the probability of becoming *excited* after a deflection, is $1/13n$.

LEMMA 6.1. *If the packet π is in the excited state, it will complete a home run with probability at least $1/4e$.*

Proof. By Theorem 5.1, the probability of a home run is at least $\frac{1}{2}(1-p)^{12n+m}$. With $p = 1/13n$ and $m = n$, and by applying Equation 6.1 we have

$$\begin{aligned}\frac{1}{2}(1-p)^{12n+m} &= \frac{1}{2}\left(1-\frac{1}{13n}\right)^{13n}\\ &\geq \frac{1}{2}\frac{1}{e}\left(1-\frac{1}{13n}\right)\\ &\geq \frac{1}{2}\frac{1}{2e}.\end{aligned}$$

LEMMA 6.2. *Each time a packet is deflected, it completes a home run to its destination with probability at least $1/52en$.*

Proof. Let π be a deflected packet that gets *excited* with probability p. According to Lemma 6.1 it will complete a home run with probability at least $1/4e$. Therefore, the probability for completing a home run after a deflection is $p \cdot 1/4e = 1/52en$.

LEMMA 6.3. *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 from π to its destination is no more than $2n-2$. Each time π is deflected, the distance increases, and each time it follows a good link, it decreases.

We now show that the expected number of steps for a single packet to reach its destination is optimal.

THEOREM 6.1. *The expected number of steps for a packet to arrive at its destination is $O(n)$.*

Proof. Let $q = 1/52en$ be the probability to do a home run after a deflection, such as calculated in Lemma 6.2. The expected number of deflections is given by

$$\begin{aligned}\sum_{x=1}^{\infty} x \cdot \Pr[\text{exactly x deflections}] &= \sum_{x=1}^{\infty} x \cdot q \cdot (1-q)^{x-1}\\ &= \frac{q}{(1-(1-q))^2}\\ &= \frac{1}{q}.\end{aligned}$$

Therefore, the expected number of deflections is $52en$. By Lemma 6.3, we have an expected number of $2(52en) + 2n - 2$ steps.

7 Analysis "with High Probability"

In this section we show that with a proper choice of p, all packets will be delivered within time $O(m \ln n)$ with high probability (meaning with probability at least $1 - 1/n$). The challenge here is that we assume that nodes do not know m, the instance-specific measure of congestion. The key technique here is to allow p to vary with time, ensuring that p lies within the "right" range sufficiently long to guarantee timely delivery. If the value of m were known to the algorithm, p could be constant (and approximately $1/m$).

We will use the following constants.

$$c = 36e \qquad c' = 39c$$
$$t_0 = c'm \ln n + 2n \qquad t_1 = 3c'm \ln n$$

Let the probability p that a deflected normal packet becomes *excited* be the following function of time:

$$p(t) := c \ln t / t.$$

Notice that when t lies in the range t_0 to t_1, the value of $p(t)$ is approximately $1/m$, the desired value.

LEMMA 7.1. *If packet π becomes excited at time $t \geq t_0$, then the probability of completing a home run is at least $1/4e$.*

Proof. Any packet conflicting with π must have started its home run attempt at most $2n$ steps before π. The probability that such a packet became *excited* was $p' \leq p(t_0 - 2n) = c \ln(c'm \ln n)/(c'm \ln n)$. By Theorem 5.1, the probability of a home run is at least $\frac{1}{2}(1-p')^{12n+m}$. Since $n \leq m \leq 4n^2$, by taking n to be sufficiently large that $4c' \ln n \leq n$, and by applying Equation 6.1 we have

$$\begin{aligned} \frac{1}{2}(1-p')^{12n+m} &\geq \frac{1}{2}\left(1 - \frac{c \ln(c'm \ln n)}{c'm \ln n}\right)^{13m} \\ &> \frac{1}{2}\left(1 - \frac{3c \ln n}{c'm \ln n}\right)^{13m} \\ &\geq \frac{1}{2}\left(1 - \frac{1}{13m}\right)^{13m} \\ &\geq \frac{1}{2}\frac{1}{e}\left(1 - \frac{1}{13m}\right) \\ &\geq \frac{1}{2}\frac{1}{2e}. \end{aligned}$$

LEMMA 7.2. *Each time t (with $t_0 \leq t \leq t_1$) a packet is deflected, it will complete a home run with probability at least $c/12ec'm$.*

Proof. The probability of getting excited is at least

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

The probability of completing a home run when getting excited is according to Lemma 7.1 at least $1/4e$. Therefore, the probability of having a home run when being deflected is at least $c/3c'm \cdot 1/4e = c/12ec'm$.

LEMMA 7.3. *With probability at least $1 - 1/n^3$, a packet will reach its destination in t_1 steps.*

Proof. By Lemma 7.2, each time that a packet π is deflected in the time interval $[t_0, t_1]$, packet π will complete a home run with probability at least $c/12ec'm$. Because the adversary is allowed to redistribute the other packets at each deflection, successive probabilities are independent. By Lemma 6.3, the number of deflections that can fit in the time interval $t_0 \leq t \leq t_1$ is at least $x = (t_1 - t_0 - 2n)/2 = m \ln n(3c' - c')/2 = c'm \ln n$. Therefore, π will fail to reach its destination after x deflections with probability at most $(1 - c/12ec'm)^x$. By Equation 6.1 we have

$$\begin{aligned} \left(1 - \frac{c}{12ec'm}\right)^x &\leq \left(1 - \frac{3}{c'm}\right)^{c'm \ln n} \\ &\leq e^{-3 \ln n} \\ &= 1/n^3. \end{aligned}$$

THEOREM 7.1. *With high probability, all packets reach their destination nodes in at most $O(m \ln n)$ steps.*

Proof. By Lemma 7.3, 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 packets (at most n^2) will arrive at their destinations within t_1 steps is at least (by applying Equation 6.2)

$$\begin{aligned} \left(1 - \frac{1}{n^3}\right)^{n^2} &= \left(1 - \frac{1/n}{n^2}\right)^{n^2} \\ &\geq 1 - \frac{1}{n}. \end{aligned}$$

8 Applications

For the batch permutation problem, $m = n$, and for the random destination problem, it can be shown that $m = O(n)$ with high probability. From Theorem 7.1, we have the following corollary.

COROLLARY 8.1. *For the batch permutation or random destination problem, with high probability, all packets reach their destination nodes in at most $O(n \ln n)$ steps.*

Consider the following *rectangle routing problem.* There are n^2 packets whose destinations are distributed uniformly within a $w \times h$ rectangle, and all packets originate outside the rectangle. Uniform distribution means that every node within the rectangle is the destination of $\Theta(n^2/wh)$ packets. Assume (WLOG) that $w \geq h$. Theorem 7.1 says that our algorithm finishes in $O(h \cdot n^2/wh \cdot \ln n) = O(n^2/w \cdot \ln n)$ steps with high probability.

Mansour and Patt-Shamir [15] have noted that there is a trivial lower bound for problems of this kind: $\Omega(d_{max} + W)$, where d_{max} is the maximum distance any packet must traverse, and W is the *network bandwidth lower bound* (defined as the maximum, over all node subsets S, of the number of packets with destination in S divided by the number of links leading to S from nodes not in S. For our $w \times h$ size rectangle routing problem, $W = n^2/(2w + 2h) = \Theta(n^2/w)$. Thus with high probability our algorithm is $O(\ln n)$-competitive with the trivial lower bound for this class of problems.

9 Discussion

Our results apply almost verbatim to the $n \times n$ torus. The only difference is that distances are smaller on a torus: any two nodes are at most n links apart instead of $2n$. As a result, some of the constants are smaller for the torus, reducing the complexity measures by a constant factor.

One important open problem is how to analyze dynamic problems, where packets are inserted into the network at a steady rate (not just at time zero). We think that techniques similar to those proposed by Broder and Upfal [8] are promising.

An interesting issue related to the rectangle routing problem concerns problems in which destinations are assigned in a non-uniform way.

We have been using the value m as an estimator for the inherent difficulty of a batch routing problem. The relation of m with the known lower bound $\Omega(d_{max} + W)$ remains imperfectly understood. It would be interesting to derive an algorithm whose complexity was expressed directly in terms of the lower bound instead of m.

Acknowledgments

We are grateful to Gopal Panduragan, Eli Upfal, and the referees.

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] A. Bar-Noy, P. Raghavan, B. Schieber, and H. Tamaki. Fast deflection routing for packets and worms. In *Proceedings of the Twelth Annual ACM Symposium on Principles of Distributed Computing*, pages 75–86, Ithaca, New York, USA, August 1993.

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

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

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

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

[7] 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.

[8] A. Broder and E. Upfal. Dynamic deflection routing on arrays. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, pages 348–358, May 1996.

[9] 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, October 1992. IEEE Computer Society Press.

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

[11] B. Hajek. Bounds on evacutation time for deflection routing. *Distributed Computing*, 1:1–6, 1991.

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

[13] Ch. 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.

[14] 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.

[15] 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.

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

[17] I. Newman and A. Schuster. Hot-potato algorithms for

permutation routing. *IEEE Transactions on Parallel and Distributed Systems*, 6(11):1168–1176, November 1995.

[18] A. Schuster. *Bounds and Analysis Techniques for Greedy Hot-Potato Routing*, chapter 11, pages 283–354. Optical Interconnections and Parallel Processing: The Interface. Kluwer Academic Publishers, 1997.

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

[20] 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.

[21] 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, September 1997.

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

[23] 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.