Randomized Routing with Shorter Paths

Eli Upfal, Sergio Felperin, and Marc Snir, *Fellow, IEEE*

Abstract—We study in this paper the use of randomized routing in multistage networks. While log N additional randomizing stages are needed to break "spatial locality," within each permutation, only log log N additional randomizing stages are needed to break "temporal locality" among successive permutations. Thus, log N bits of initial randomization per input, followed by log log N bits of randomization per packet are sufficient to ensure that t permutations are delivered in time t + log N. We present simulation results that validate this analysis.

Index Terms—Butterfly networks, interconnection networks, multistage networks, packet-switching, parallel communication, randomized routing.

1 Introduction

The problem of routing a large number of messages simultaneously through a sparse communication network is fundamental to the theory and practice of parallel computing. This problem has been widely studied for a variety of network topologies and routing algorithms. In particular, much attention has been given to the advantage that accrues from the use of randomized routing. Our paper attempts to further clarify the role of randomization in routing.

Consider a multicomputer system consisting of N processors connected via a multistage message switching network, with a Butterfly topology: The network consists of $\log N$ stages of 2×2 switches, and network inputs are identified with network outputs. (We use this particular topology for sake of concreteness; similar results hold for related networks, such as hypercubes or fat trees. See [5] or [12] for more information on multistage networks.) It is well known that a random permutation can be routed through such network in $O(\log N)$ steps [1], [16], [15]. On the other hand, a worse-case permutation, such as transpose, requires $\Omega(\sqrt{N})$ steps to be routed. This situation can be remedied by adding extra stages to the network, and randomizing routing through these extra stages. If log N stages are added, then any permutation will be routed in time $O(\log N)$, with high probability. This is, essentially, the strategy adopted for routing on the CM-5 [6]. The cost is significant: the network size is doubled or, equivalently, the network bandwidth is halved.

In practice, one is seldom interested in the time needed to route one permutation. Rather, one pipelines several communication steps, so as to amortize the Ω (log N) transfer time through the network. If log N extra randomizing stages are used, then t consecutive permutations can be routed optimally in time $O(t + \log N)$, with high probability [16].

Can we achieve the same effect at less cost? It was shown in [8] that fewer randomizing stages will not do; if less than $(\log N)/2$ randomizing stages are used, then a worst case permutation requires $\Omega(\sqrt{N})$ to be routed; t successive executions of that same worst case permutation will require $\Omega(t\sqrt{N})$ steps to be routed.

Alternatively, one might save by using some of the additional randomizing stages less frequently. Consider the following practical alternative. Before the computation starts, processors are randomly permuted, i.e., logical addresses are assigned randomly to physical processors. This is easily done at start-up time, assuming that all processors are identical: One assigns processes to processors in a random manner. This, essentially, provides the same amount of randomization as log N randomizing network stages. Indeed, we can show that each permutation takes $O(\log N)$ steps to route, with high probability (Theorem 2).

What happens now if we pipeline the transmission of t successive permutations? If the permutations are independent, then t successive permutations can be routed optimally in $O(t + \log N)$ steps, with high probability. However, this is not true if the successive permutations are dependent, in particular, if the same permutation is executed repeatedly. We show that, for some permutations, the expected number of steps required to route the same permutation t times is Ω ($t \log N/\log \log N$) (Corollary 5).

The randomization of node addresses takes care of "spatial dependency," i.e., the correlation between destinations of distinct messages within one permutation. It does not take care of "temporal dependency," i.e., the correlation between successive permutation steps. In practice, one can expect temporal dependency to be a significant effect. Consider, for example, a computation that emulates a fixed topology algorithm, e.g., a grid algorithm where all communications are horizontal or vertical shifts. Then communication consists of a fixed number of permutations that are repeatedly executed.

- *E. Upfal is with IBM Almaden Research Center, San Jose, California; and The Weizmann Institute of Science, Rehovot, Israel.*
- *S. Felperin is with IBM System Architecture Group, Argentina.*
- *M. Snir is with IBM T.J Watson Research Center, Yorktown Heights, New York. E-mail: Snir@watson.ibm.com.*

Manuscript received June 10, 1993; revised Jan. 20, 1995.
For information on obtaining reprints of this article, please send e-mail to: transactions@computer.org, and reference IEEECS Log Number D95089.

"Bulk" randomization, done once per computation task, is not sufficient. Some randomization needs to be introduced at each permutation task. However, one need not use log N randomization stages. We analyze the following strategy: First, processor addresses are randomly permuted; then log log N randomizing stages are used to route each permutation. We show that this is sufficient to optimally route t permutations in time $O(t + \log N)$ with high probability (Corollary 12).

Finally, we give results of simulations of the effect of adding extra randomizing stages to a Butterfly network. As predicted by the analysis, latency decreases when a few (log log N) extra stages are added. However, when more stages are added, the average latency increases back, since the decrease in congestion is offset by the increase in network depth. Thus, within the limitations of the communication model we study, there is an optimal number of additional randomizing stages that need be added to a Butterfly network: too much or too little randomization both hurt. The simulation shows that the constants implied in the analysis are small enough for the scheme to be practical.

Multistage networks with few additional stages have been studied in the past by several authors, mostly in the context of circuit switching. In particular, we would like to note the results of Mitra and Cieslak [7].

2 Routing in a Butterfly

We show in this section that "one-time" randomization, done before the execution of the computation task, is not sufficient for efficient communication. We consider the following scenario: Processes are assigned to random processors before execution, routing is done deterministically, through a Butterfly with no extra stages. We prove in this section that routing one arbitrary permutation takes, with high probability, optimal $O(\log N)$ parallel steps (Theorem 2). However, routing a sequence of t arbitrary permutations in that scenario takes with high probability $\Omega(t \log N/\log \log N)$ parallel steps, rather than optimal $O(t + \log N)$ time. (Corollary 5).

In the next section we show that the optimal $O(t + \log N)$ time result can be achieved with high probability, using limited "run-time" randomization,

An N Butterfly is an acyclic layered network with $N = 2^n$ inputs, N outputs, and $n - 1$ internal stages, each with N nodes. We number the stages of the networks by $0, \ldots, n$, and the nodes in each stage by $0, \ldots, N - 1$. Let v be a node in stage s, and let $(a_1, \ldots, a_n)$ be the binary presentation of its number in that stage. We denote the number of a node v by $(s, (a_1, \ldots, a_n))$. Node $(s - 1, (a_1, \ldots, a_n))$ is connected to two nodes in stage s, specifically to nodes with numbers $(s, (a_1, \ldots, a_n))$ and $\left(s, \left(a_1, \ldots, \bar{a}_s, \ldots, a_n\right)\right)$, where $\bar{a}_s$ is the complement of a_s. Input $(0, (a_1, \ldots, a_n))$ is identified with output $(n, (a_1, \ldots, a_n))$. There is a unique path from each input $(a_1, \ldots, a_n)$ to each output $(b_1, \ldots, b_n)$, namely, the path with nodes $(0, (a_1, \ldots, a_n))$, $(1, (b_1, a_2, \ldots, a_n))$, $(2, (b_1, b_2, a_3, \ldots, a_n))$, $\ldots$, $(n, (b_1, b_2, \ldots, b_n))$.

We consider the following greedy routing algorithm: Packets are routed on the unique path from source to destination. Each packet is assigned upon entry a random priority uniformly chosen in a range $1, \ldots, K$. ($K = \Theta(\log N + p)$, where p is the maximum number of packets at any source or destination.) Each input and switch forward a packet at each cycle, whenever a packet is queued at the node. Conflicts are resolved by forwarding the packet with highest priority, with ties broken according to the packet source. (Priorities are an artifact needed for proving upper bounds. In practice, random conflict resolution seems adequate.)

We define $C(\pi, v)$, the *congestion of permutation π at node v*, to be the number of packets that are routed through node v in the execution of this permutation. (Congestion does not depend on the priorities assigned to packets, since routes are fixed.) $C(\pi)$, the *congestion of permutation π* is defined to be

$$C(\pi) = \max_v C(\pi, v).$$

This definition is extended to routing problems that are not necessarily permutations. Clearly, $C(\pi)$ is a lower bound on the time needed to route permutation π, for any routing algorithm. On the other hand, the following result holds [5]:

THEOREM 1. *With probability $\geq 1 - 1/N^7$ the greedy routing algorithm executes any routing problem with congestion $C \geq \log N$ on the Butterfly in time $O(C)$.*

A random renaming of the inputs/outputs of the network replaces each permutation π with the permutation $\sigma^{-1} \pi \sigma$, σ random.

THEOREM 2. *Let $\pi \in S_N$ be a fixed permutation. Let $t(\sigma^{-1}\pi\sigma)$ be the number of steps used by the greedy routing algorithm to route the permutation $\sigma^{-1}\pi\sigma$. Then there exists a constant "a" such that*

$$\Pr\{t(\sigma^{-1}\pi\sigma) < a \log N\} > 1 - 1/N$$

for random σ.

PROOF. It is well known that if π is a uniformly chosen random permutation, then with high probability π can be routed on the Butterfly in $O(\log N)$ steps [5]. The difficulty in proving our theorem is that the permutation $\sigma^{-1}\pi\sigma$ is not a random permutation chosen uniformly from among all permutations. In particular, $\sigma^{-1}\pi\sigma$ has the same cycle structure as the permutation π.

By Theorem 1, it is sufficient to show that each node of the Butterfly is visited by at most $O(\log N)$ packets with probability $< 1/N$.

Let I be the set of entries i such that $\pi(i) = i$, and let π_0 be the restriction of π to I. Then $\sigma^{-1}\pi_0\sigma$ is the identity mapping on the set $\sigma(I)$, and no more than one packet traverses each node in the routing of this subset of the permutation $\sigma^{-1}\pi\sigma$.

Let π' denote the permutation $\pi - \pi_0$. Consider the graph that represents π' (with vertices $\{0, \ldots, N - 1\}$ and edges $(i, \pi'(i))$). This graph has degree ≤ 2 and has no self loops, thus its edges can be colored by 3 colors so that no vertex is incident to two edges of the same color. Let π_1, π_2, and π_3 denote a partition of the permutation π' that corresponds to such a coloring.

Let $\{x_1, ..., x_m\}$ be the domain of π_1 and let $y_i = \pi(x_i)$, $i = 1, ..., m$. Then $\{x_1, ..., x_m\} \cap \{y_1, ..., y_m\} = \emptyset$, so that $m \le N/2$. Let v be a node at level s, $IN(v)$ be the set of 2^s inputs that reach node v, and $OUT(v)$ be the set of 2^{n-s} outputs that can be reached from v. Each path from an input in $IN(v)$ to an output in $OUT(v)$ traverses node v. The permutation $\sigma^{-1}\pi_1\sigma$ maps $\sigma(x_i)$ to $\sigma(y_i)$, $I = 1, ..., m$. Therefore $C(\sigma^{-1}\pi_1\sigma, v)$, the congestion at node v caused by the permutation $\sigma^{-1}\pi_1\sigma$, equals the number of indices j, $1 \le j \le m$, such that $\sigma(x_j) \in IN(v)$ and $(y_j) \in OUT(v)$.

Let $\{u_1, ..., u_N\}$ be some ordering of the Butterfly inputs. A random permutation σ can be generated by the following process: $\sigma(u_1)$ is chosen uniformly at random from the set $\{0, ...N, -1\}$. If $\sigma(u_1), ..., \sigma(u_{i-1})$ have been chosen then $\sigma(u_i)$ is chosen uniformly at random from the set $\{0, ...N-1\} - \{\sigma(u_1), ..., \sigma(u_{i-1})\}$. It is easy to see that this process generates a random permutation chosen uniformly from S_N, regardless of the order of $u_1, ..., u_N$. We shall use this process, with inputs ordered in a sequence $x_1, ..., x_m$, followed by the remaining inputs.

There are $\binom{2^{n-s}}{r}$ possible choices of r indices j such that $\sigma(y_j) \in Out(v)$. For each $j \le m$ there are at least $N/2$ distinct choices for $\sigma(x_j)$. Thus, $\Pr\{\sigma(x_j) \in IN(v)\} \le 2^s/(N/2)$, independently of the choices made for other inputs in $\{x_1, ..., x_m\}$. Therefore,

$$\Pr\left\{C\left(\sigma^{-1}\pi_1\sigma, v\right) \ge r\right\} \le \binom{2^{n-s}}{r}\left(\frac{2^s}{N/2}\right)^r$$

We use the inequality $r! > (r/e)^r$ to bound the last expression by

$$\left(\frac{2^{n-s}e}{r}\right)^r\left(\frac{2^s}{N/2}\right)^r = \left(\frac{2e}{r}\right)^r.$$

Thus, the probability that $C(\sigma^{-1}\pi_1\sigma) \ge \log N$ can be bounded by

$$N \log N\left(\frac{2e}{\log N}\right)^{\log N} \le \frac{1}{4N},$$

for sufficiently large N. The same argument applies to the permutations π_2 and π_3. Thus,

$$\Pr\left\{C\left(\sigma^{-1}\pi\sigma\right) \ge 1 + 3\log N\right\} \le \tfrac{3}{4N},$$

and the theorem follows. □

THEOREM 3. *There exists a constant "a" such that*

$$\Pr\left(C(\pi) > a \log N/\log\log N\right) > 1 - 1/N$$

for a random permutation π.

PROOF. Let $b(x)$ denote the $n/2$ digit binary representation of the number x, $x \le \sqrt{N}$. Consider the set of $\ell = \sqrt{N}/16$ nodes in stage $n/2$ with numbers $(n/2, (b(x), b(x)))$, $x = 0, ..., \ell - 1$. Let $IN(x)$ denote the set of $\sqrt{N}$ inputs that can reach node $(n/2, (b(x), b(x)))$ and let $OUT(x)$ denote the set of $\sqrt{N}$ outputs that can be reached from this node. Note that the sets $IN(x)$ are disjoint.

We generate a random permutation π by first choosing the mappings of the inputs in the sets $IN(x)$, $x = 0, ..., \ell - 1$. Let $\mathcal{E}$ be the event: "No set $OUT(x)$, $x = 0, ..., \ell - 1$, is a mapping of more than $\sqrt{N}/4$ inputs in $\cup_{i=0}^{\ell-1} IN(i) - IN(x)$". Let $\mathcal{E}(x)$ be the event "$a \log N/\log\log N$ packets from $IN(x)$ are sent to outputs in $OUT(x)$."

Note that the event $\mathcal{E}(x)$ implies that $a \log N/\log\log N$ packets traverse the node $(n/2, b(x), b(x))$.

If the event $\mathcal{E}$ holds, and less than $a \log N/\log\log N$ elements $u \in IN(x)$, $u < v$, have been mapped to $OUT(x)$, then $v \in IN(x)$ has the choice of at least $\sqrt{N} - \sqrt{N}/4 - a\log N/\log\log N > \sqrt{N}/2$ outputs in $OUT(x)$. Hence, the probability that v is mapped to $OUT(x)$ is at least $\frac{\sqrt{N}/2}{N} = \frac{1}{2\sqrt{N}}$, independently of the choices made for previous inputs in $IN(x)$, and independently from choices made for inputs in $IN(y)$, $y < x$. Thus,

$$P\{\mathcal{E}(x) \mid \mathcal{E}, \neg\mathcal{E}(0), ..., \neg\mathcal{E}(x-1)\}$$

$$\ge \left(\frac{\sqrt{N}}{\frac{a\log N}{\log\log N}}\right)\left(\frac{1}{2\sqrt{N}}\right)^{\frac{a\log N}{\log\log N}}\left(1 - \frac{1}{2\sqrt{N}}\right)^{\sqrt{N} - \frac{a\log N}{\log\log N}}$$

But

$$\left(1 - \frac{1}{2\sqrt{N}}\right)^{\sqrt{N} - \frac{a\log N}{\log\log N}} > \left(1 - \frac{1}{2\sqrt{N}}\right)^{\sqrt{N}} > \frac{1}{2}$$

and,

$$\left(\frac{\sqrt{N}}{\frac{a\log N}{\log\log N}}\right)\left(\frac{1}{2\sqrt{N}}\right)^{\frac{a\log N}{\log\log N}} >$$

$$\left(\frac{\sqrt{N} - a\log N/\log\log N}{a\log N/\log\log N}\right)^{\frac{a\log N}{\log\log N}}\left(\frac{1}{2\sqrt{N}}\right)^{\frac{a\log N}{\log\log N}}$$

$$> \left(\frac{1}{\log N}\right)^{\frac{a\log N}{\log\log N}} = N^{-a},$$

for large enough N. Thus,

$$\Pr\left\{\mathcal{E}(x) \middle| \mathcal{E}, \neg\mathcal{E}(0), \ldots, \neg\mathcal{E}(x-1)\right\} > \frac{N^{-a}}{2},$$

and

$$\Pr\left\{\bigcap_{x=0}^{\ell-1} \neg\mathcal{E}(x) \middle| \mathcal{E}\right\} = \Pr\left\{\neg\mathcal{E}(0) \middle| \mathcal{E}\right\}\Pr\left\{\neg\mathcal{E}(1) \middle| \mathcal{E}, \neg\mathcal{E}(0)\right\}...$$

$$...\Pr\left\{\neg\mathcal{E}(\ell) \middle| \mathcal{E}, \neg\mathcal{E}(0), \ldots, \neg\mathcal{E}(\ell-1)\right\} \le \left(1 - \frac{N^{-a}}{2}\right)^{\sqrt{N}/16}.$$

To bound the probability of $\neg\mathcal{E}$, recall that we are choosing first the mappings of the $N/16$ inputs in $IN(x)$, $x = 0, .., \ell - 1$. There are at least $15N/16$ distinct choices for the mapping of each input, out of which at most $\sqrt{N}$ are in $OUT(x)$. Thus, the probability that the set $OUT(x)$ receives more than $\sqrt{N}/4$ packets from inputs in $\bigcup_{i=0}^{\ell-1} IN(i) - IN(x)$ is bounded by

$$\binom{N/16}{\sqrt{N}/4}\left(\frac{\sqrt{N}}{15N/16}\right)^{\sqrt{N}/4} < \left(\frac{eN/16}{\sqrt{N}/4}\right)^{\sqrt{N}/4}\left(\frac{\sqrt{N}}{15N/16}\right)^{\sqrt{N}/4} = \left(\frac{4e}{15}\right)^{\sqrt{N}/4}.$$

Therefore,

$$Pr(\neg\mathcal{E}) < \frac{\sqrt{N}}{16}\left(\frac{4e}{15}\right)^{\sqrt{N}/4},$$

and

$$Pr\left\{\bigcap_{x=0}^{\ell-1}\neg\mathcal{E}(x)\right\} \le Pr\left\{\bigcap_{x=0}^{\ell-1}\neg\mathcal{E}(x)\middle|\mathcal{E}\right\} + Pr\{\neg\mathcal{E}\}$$

$$\le \left(1-\frac{N^{-a}}{2}\right)^{\sqrt{N}/16} + \frac{\sqrt{N}}{16}\left(\frac{4e}{15}\right)^{\sqrt{N}/4} \le 1/N$$

for $a < 1/2$ and sufficiently large. □

COROLLARY 4. *There exists a permutation π such that*

$$\Pr\{C(\sigma^{-1}\pi\sigma) > a \log N/\log\log N\} > 1 - 1/N$$

for random σ.

PROOF. For a random permutation π, the permutation $\sigma^{-1}\pi\sigma$ is a uniformly distributed random permutation. □

COROLLARY 5. *There exists a permutation π such that routing $\sigma^{-1}\pi\sigma$ t times requires at least at $\log N/\log\log N$ parallel steps, with probability $> 1 - 1/N$, for random σ.*

Since the last lower bound is derived from a lower bound on congestion, it holds for any routing algorithm (i.e., for any conflict resolution scheme).

3 Routing on a Butterfly with Extra Stages

We prove in this section that there exists a set of almost all permutations (more than $N!\left(1-\frac{1}{N^c}\right)$ permutations), such that if an adversary picks a sequence of t permutations from this set (possibly with repetitions), our algorithm routes all these permutations, with high probability, in asymptotically optimal time. Furthermore, if processor numbers are randomly permuted then any polynomially large set of permutations belong to this "good" set with high probability.

We use a Butterfly with $r \ge \log\log N$ extra stages. The first $n + 1$ stages of our network is a regular Butterfly, the extra r stages repeat the first r stages of the network.

Our routing algorithm starts with r random transitions followed by n deterministic transitions. A packet at an input node chooses, randomly and independently of the other packets, one of the 2^r nodes that it can reach at stage r. In its first r transitions the packet proceeds to that node. In the following n transitions the packet proceeds to its destination output using the unique path from the nodes it reached in stage r to its destination. i.e., a packet with source $(a_1, ..., a_n)$ and destination $(b_1, ..., b_n)$ traverses the path $(0, (a_1, ..., a_n)), (1, (x_1, a_2, ..., a_n)),...,(r, (x_1, ..., x_r, a_{r+1}, ..., a_n)), (r + 1, (x_1, ..., x_r, b_{r+1}, ..., a_n)), ..., (n, (x_1, ..., x_r, b_{r+1}, ..., b_n)), (n + 1, (b_1, ..., x_r, b_{r+1}, ..., b_n)), ..., (n + r, (b_1, ..., b_r, b_{r+1}, ..., b_n))$, where $x_1, ..., x_r$ are randomly chosen.

Queues in intermediate nodes are ordered according to priorities given to the packets in the inputs. A packet in permutation ℓ in the sequence receives priority $\left(\left\lceil\frac{\ell}{\log N}\right\rceil, R\right)$ where R is a random number uniformly chosen in the range $\left[0,\ldots,c(\log N)2^r\right]$. The priorities are ordered in lexicographic order. The priorities have two roles: The deterministic, more significant, part of the priority guarantees that permutations arrive "in order." The random part of the priority is a technical requirement for analyzing the algorithm (see [5] for alternative *nonpredictive* queue policies).

3.1 A "Good" Set of Almost All Permutations

Let $B(s, (a_{r+1}, ..., a_n))$ denote the set of 2^r nodes in column s of a Butterfly (with no extra stages) with the $n - r$ rightmost bits in their addresses equal to $(a_{r+1}, ..., a_n))$, i.e.,

$$B(s, (a_{r+1}, ..., a_n)) = \{(s, (x_1, ..., x_r, a_{r+1}, ..., a_n) \mid x_i = 0, 1\}.$$

Let π be an arbitrary N–permutation. Consider the routing of π on a Butterfly with no extra stages using the unique paths. Let $M(\pi, s, (a_{r+1}, ..., a_n))$ denote the set of packets traversing nodes in $B(s, (a_{r+1}, ..., a_n))$ in routing the permutation π.

Let

$$\Psi(r, \tau) = \{\pi \mid |M(\pi, s, (a_{r+1}, ..a_n))| < \tau \text{ for all } 0 \le s \le n \text{ and all } (a_{r+1}, ..a_n)\}.$$

THEOREM 6.

$$\left|\Psi\left(r, c2^r\right)\right| \ge N!\left(1-\frac{1}{e^{c2^r}}\right) > N!\left(1-\frac{1}{Nc}\right),$$

for $r \ge \log\log N$ and $c \ge 7$.

PROOF. Observe that if $s \le r$, a set $B(s, (a_{r+1}, ..., a_n))$ can be reached from 2^r inputs, thus $|M(\pi, s, (a_{r+1}, ..., a_n))| \le 2^r < c2^r$ for all π. If $s > r$ then the set $B(s, (a_{r+1}, ..., a_n))$ can be reached from 2^s inputs, and can send messages to $2^r 2^{n-s}$ outputs. Thus no more than $2^{(r+n)/2} = \max_s \min[2^s, 2^r 2^{n-s}]$ packets can pass $B(s, (a_{r+1}, ..., a_n))$ in the routing of any permutation.

There are $N/2^r$ possible sets $B(s, (a_{r+1}, ..., a_n))$ in column s. Thus we can bound the number of permutations that are not in $\Psi(r, c2^r)$ by

$$\sum_{s=r}^{n-1}\sum_{k=c2^r}^{(n+r)/2}\frac{N}{2^r}\binom{2^s}{k}\binom{2^r 2^{n-s}}{k}k!(N-k)! =$$

$$\sum_{s=r}^{n-1}\sum_{k=c2^r}^{(n+r)/2}\frac{(N-k)!N}{2^r k!}2^s\left(2^s-1\right)\ldots$$

$$\left(2^s-k+1\right)\left(2^r 2^{n-s}\right)\left(2^r 2^{n-s}-1\right)\ldots\left(2^r 2^{n-s}-k+1\right).$$

Since

$$(2^s - j)(2^r 2^{n-s} - j) \le 2^r (N - j)$$

for $j \le 2^{(n+r)/2}$, the above sum is bounded by

$$nN!\frac{N}{2^r}\sum_{k\ge c2^r}\frac{2^{rk}}{k!} = nN!\frac{N}{2^r}\frac{2^{rc2^r}}{(c2^r)!}\sum_{k\ge 0}\frac{2^{rk}}{(c2^r+1)\ldots(c2^r+k)}$$

$$\le nN!\frac{N}{2^r}\frac{2^{rc2^r}}{(c2^r)!}\sum_{k\ge 0}\frac{2^{rk}}{(c2^r)^k} = nN!\frac{N}{2^r}\frac{2^{rc2^r}}{(c2^r)!}\frac{c}{c-1}.$$

$$\le nN!\frac{N}{2^r}2^{rc2^r}\left(\frac{e}{c2^r}\right)^{c2^r}\frac{c}{c-1} = \frac{N!}{N^c}\frac{n}{2^r}\frac{c}{c-1}\left(\frac{2^{c+1}e^c}{c^c}\right)^{2^r} \le \frac{N!}{N^c}. \quad \square$$

LEMMA 7. *Let π be a fixed permutation. Then, for random σ,*

$$\Pr\left(\sigma^{-1}\pi\sigma \in \Psi(r, c2^r)\right) \ge 1 - \frac{3}{N^{c/3-1}}.$$

PROOF. Similar to the proof of Theorem 2, we partition π to four subpermutations π_1, π_2, π_3, and π_4, such that π_1 is the identity permutation on some of the elements, and in π_i, $i = 2, 3, 4$, each element appears no more than once, either as an input or as an output for a packet. Thus $\sigma^{-1}\pi_i\sigma$, $i = 2, 3, 4$, is part of a random permutation and we can apply Theorem 6 with parameter $c/3 - 1$. $\square$

3.2 Analysis of the Algorithm

We first treat the case of a sequence of up to $\log N$ permutations.

THEOREM 8. *Given any sequence of $t \le \log N$ permutations from the set $\Psi(r, c2^r)$ (possibly with repetitions), our algorithm routes all the permutations on a Butterfly with extra r stages in $c\beta\alpha \log N$ steps with probability $1 - e^{-\beta \log N}$ for some constant $\alpha > 0$ and any $\beta > 1$.*

PROOF. Consider an execution of the algorithm on a sequence of t permutations. Let C be the maximum number of packets traversing any node in the network in this execution (the congestion of this routing problem).

LEMMA 9.

1) $C \le c \log N 2^r$;
2) *Prob* $\{C \ge 6c \log N\} \le 2^{-4 \log N}$.

PROOF. We use the following version of the Chernoff Bound [2], [4] in this proof:

THEOREM 10. *Let $X_1, \ldots, X_m$ be a sequence of independent $0-1$ random variable. Let $Pr\{X_i = 1\} = p_i$, let $X = \sum_{i=1}^{m} X_i$, and let $P = \sum_{i=1}^{m} P_i$. For any $s > 6P$,*

$$Pr\{X > s\} \le 2^{-s}.$$

We distinguish between nodes in the first r stages of the network and nodes in the remaining n stages.

A node in the first r stages can receive packets from no more than 2^r inputs, thus can never receive more than $t2^r$ packets in routing t permutations. Since packets follow a random path in the first r stages, the number of packets that pass a given node is distributed $B(t2^r, 2^{-r})$. By the Chernoff bound, the probability that more than $6c \log N$ packets traverse a node in the first r stages is bounded by $2^{-6c \log N}$.

A packet p traverses a node $(s, (a_1, \ldots, a_n))$ only if has source $(b_1, \ldots, b_s, a_{s+1}, \ldots, a_n)$ and destination $(c_1, \ldots, c_r, a_{r+1}, \ldots, a_s, c_{s+1}, \ldots, c_n)$, for some choice of b_i and c_i.

Such packet would traverse the node $(s, (c_1, \ldots, c_r, a_{r+1}, \ldots, a_n) \in S(s, (a_{r+1}, \ldots, a_n))$, if routed on a Butterfly with no extra stages. Thus, only packets in $M(\pi, s, (a_{r+1}, \ldots, a_n))$ may traverse a node (s,a), $s>r$, in our routing algorithm. Since each of the t permutations is from the set $\Psi(r, c2^r)$, and $|M(\pi, s, (a_{r+1}, \ldots, a_n))| \le c2^r$ for each of these permutations, a node in stage $s>r$ can receive no more than $tc2^r \le c \log N 2^r$ packets.

A packet $p \in M(\pi, s, (a_{r+1}, \ldots, a_n))$ actually traverses node (s, a) only if the first r random transitions selected where $a_1, \ldots, a_r$. This happens with probability 2^{-r}. Thus, using the Chernoff bound, the probability that more than $6c \log N$ packets traverse a given node is bounded by $2^{-6c \log N}$. Since there are $N \log N$ nodes in the network, $Prob\{C \ge 6c \log N\} \le 2^{-4c \log N}$. $\square$

Having a deterministic and a probabilistic bounds on the congestion of the routing problem, we can now follow the proof of Theorem 1 in [5] to show that the probability that the routing algorithm fails to route all the packets of the t permutations in $c\alpha\beta \log N$ steps is bounded by $e^{-\beta \log N}$, for some $\alpha > 0$ and any $\beta > 1$. $\square$

We turn now to the analysis of routing long sequences of permutations.

THEOREM 11. *Given any set of t permutations from the set $\Psi(r, c2^r)$ (possibly with repetitions), our algorithm routes all the permutations on a Butterfly with extra r stages in $O(t + \log N)$ steps with high probability.*

Furthermore, for any $f \le t$, the first f permutations are routed in $O(f + \log N)$ steps with high probability.

PROOF. The priority mechanism partitions the t permutations into $\lceil t/\log N \rceil$ sets, each with up to $\log N$ permutations, such that the routing of packets in the ith set is only affected by packets in the first i sets.

The routing time of the ℓ set is thus bounded by the sum of the routing times of the first ℓ sets. By Theorem 8 the routing time of each set is stochastically bounded by exponential distribution with parameter $1/c\alpha$. The sum of ℓ independent random variables with exponential distribution has gamma distribution [3]. Thus, the probability that all the packets in the first ℓ set were not delivered in $O(\ell \log N)$ steps is bounded by $e^{-\theta'\ell \log n}$ for some $\theta' > 0$. Summing over $\ell = 1, \ldots, \left\lceil \frac{t}{\log N} \right\rceil$ we prove that the probability that any one of the t permutations was not delivered in $O(f + \log N)$ steps, where $f \le t$ was the place of the permutation in

the sequence of the t permutations, is bounded by $e^{-\theta \log N}$ for some $\theta > 0$. □

COROLLARY 12. *Given any sequence Π of $t = O(N^k)$ permutations (possibly with repetitions) and a random permutation σ, our algorithm routes all the permutations in the sequence $\sigma^{-1}\ \Pi\ \sigma$ on a Butterfly with extra $O(\log \log N)$ stages in $O(t + \log N)$ steps with high probability.*

Note that the probability domain for the results in this section is the product domain of random permutation of processors, random routing of packets through the first $O(\log \log N)$ stages, and random choice of packet priorities.

4 Simulation Results

Adding extra stages to the Butterfly network has a twofold effect on message latency. On one hand, extra stages increase the path length, thus might increase latency. On the other hand, randomization in the extra stages reduces congestion and thus might reduce latency.

In this section we report simulation results that explore this trade-off.

4.1 Node Model

An intermediate node in the network has a buffer of size one for each incoming link and an infinite queue for each outgoing link.

A step is the time required for a packet to traverse a link or move from a buffer of an incoming link to the queue of an outgoing link. Note that each packet needs two cycles (at least) to cross a single node.

4.2 The Experiments

Each run of the simulation is characterized by three parameters: $N = 2^n$–the number of inputs of the network; r–the number of extra stages; and p–the number of permutations fed to the network. Each run starts by choosing a random permutation σ: $N \rightarrow N$. Each input i generates p packets with destination $\sigma(i)$. The first r transitions of each packet are random, the last n transitions take the packet to its destination by the unique path in the last n stages of the network. The latency of a packet is the time that spans between the initiation of the experiment and the delivery of the packet to its destination. Each run was repeated 10 times with different seeds for the random generator. Each point in the graphs represents the average of the 10 independent experiments. Pseudorandom numbers were generated using a modified version of the algorithm in [10].

4.3 The Results

Figs. 1 and 2 show the effect of adding 0–12 extra stages to a Butterfly with 4K inputs. The curves represent routing of 1, 10, 20, 50, 100, and 200 identical permutations. The graphs clearly show that extra stages improve the latency in pipelining a long sequence of permutation, but do not significantly decrease the latency in routing short sequences. Furthermore, even when a large number of permutations are pipelined together, extra stages are beneficial only up to some limit. Above that limit, the decrease in congestion is offset by the increase in network depth.

Similar experiments were run for networks of size 1K, 2K, and 8K. When the data from these experiments is fitted as a function of $n, r, p, \frac{p}{2^r}$, and $\frac{pn}{2^r}$, the expressions for the average latency L_{avg}, and the maximum latency L_{max} best fit with the following coefficients:

$$L_{avg}(n,r,p) = -12.90 + 3.18n + 0.75p + 0.69\frac{p}{2^r} + 0.07\frac{np}{2^r} + 3.20r$$

$$L_{\max}(n,r,p) = -29.69 + 8.09n + 1.83p + 0.84\frac{p}{2^r} + 0.76\frac{np}{2^r} - 1.43r.$$

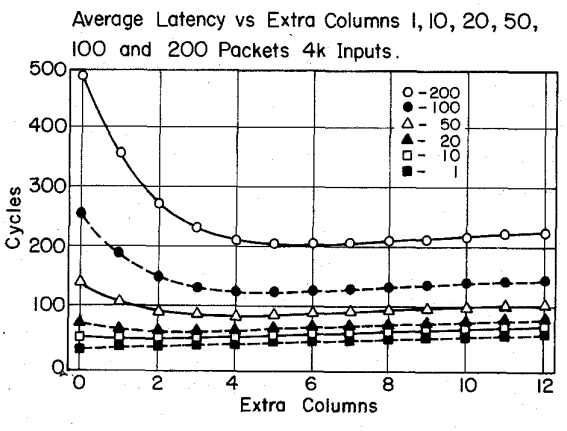
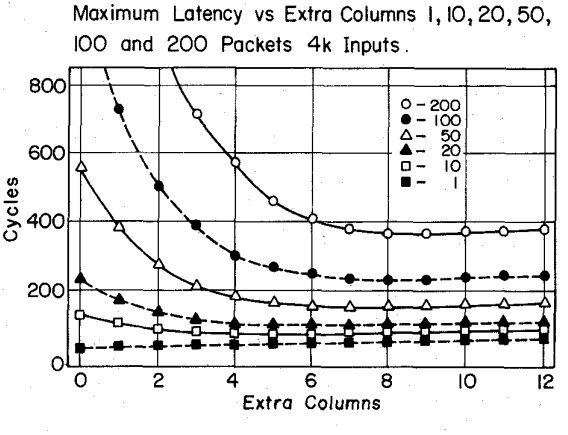


Fig. 1. Average latency vs extra columns 1, 10, 20, 50, 100, and 200 packets 4k inputs.



Maximum latency vs extra columns 1, 10, 20, 50, 100, and 200 packets 4k inputs.

5 Conclusion

One might expect intuitively that "one-time" randomization done by assigning tasks randomly to processes is as effective as "per communication" randomization that uses additional routing stages. Indeed, one-time randomization guarantees that any fixed permutation will be routed in time $O(\log N)$ with high probability (Theorem 2). However, it does not guarantee that a sequence of t permutations can be routed in time $O(t + \log N)$. Indeed, Theorem 3 shows

that the best one can expect is $O(t \log N / \log \log N + \log N)$.

At the other extreme, the result of [8] seems to imply that $\Omega(\log N)$ additional randomizing stages are needed in order to avoid congestion in a multistage network, and that the randomizing routing algorithms of [14], [15], [16], [9], [11], which at least double the number of stages, are optimal, up to a constant factor. However, the lower bound of [8] does not hold if "one-time" randomization is allowed. Indeed, since one-time randomization reduces congestion to $O(\log N / \log \log N)$, one might expect that $O(\log \log N)$ extra randomizing stages are now sufficient to avoid congestion. The main theorem of this paper shows this to be true.

The results in this paper are of more than theoretical interest. Interconnection networks for parallel machines most frequently use oblivious routing, in order to simplify the routing logic. Some redundancy is usually provided, for fault tolerance, so that more than one path connects each input to each output. In a multistage network, such redundancy can be provided by additional stages. On the other hand, fault tolerance does not justify doubling the number of stages in a multistage network: Many fewer additional stages will be sufficient, with reasonable error rates. For example, the multistage interconnection networks of the IBM SP1 and SP2 parallel computers have (at least) four distinct paths from each input to each output [13]. Our result shows that the limited redundancy in the interconnection network of a machine such as the SP1/SP2 is sufficient in avoiding congestion, with high probability.

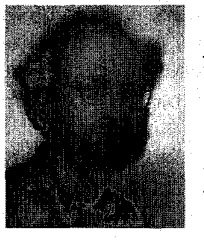
Acknowledgments

We wish to thank Tom Leighton for referring us to Theorem 3.26 in [5]. This theorem significantly simplified the proof of our main result.

Research at the Weizmann Institute was supported in part by The Norman D. Cohen Professorial Chair of Computer Science, by the Minerva Foundation, and by a grant from the Israeli Academy of Science.

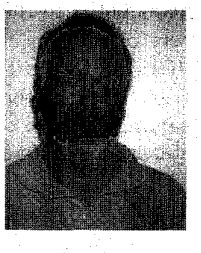
References

[1] R. Aleliunas, "Randomized Parallel Communication," *ACM-SIGOPS Symp. Principles of Distributed Systems*, pp. 60–72, 1982.
[2] H. Chernoff, "A Measure of Asymptotic Efficiency for Tests of a Hypothesis Based on the Sum of Observations," *Annals of Math. Stat.*, vol. 23, pp. 493–509, 1952.
[3] W. Feller, *An Introduction to Probability Theory and its Applications*, vol. 2. John Wiley & Sons, 1966.
[4] T. Hagerup and C. Rüb, "A Guided Tour of Chernoff Bounds," *Information Processing Letters*, vol. 33, pp. 305–308, 1989/90.
[5] F.T. Leighton, *Introduction to Parallel Algorirthms and Architectures: Arrays, Tress, Hypercubes*. San Mateo, Calif.: Morgan Kaufmann Publishers, 1992.
[6] C.E. Leiserson, et al, "The Network Architecture of the Connection Machine CM 5," *Fourth Ann. ACM Symp. Parallel Algorithms and Architectures*, pp. 272-285, 1992.
[7] D. Mitra and R.A. Cieslak, "Randomized Parallel Communications on an Extension of the Omega Network," *J. ACM*, vol. 34, pp. 802–824, 1987.
[8] P. Peleg and E. Upfal, "A Time-Randomness Tradeoff for Oblivious Routing," *SIAM J. Computing*, vol. 19, pp. 256–266, 1990.
[9] N. Pippenger, "Parallel Communication with Limited Buffers," *25th Ann. Symp. Foundations of Computer Science*, pp. 127–136, 1984.
[10] W.H. Press, *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge Univ. Press, 1988.
[11] A. Ranade, "How to Emulate Shared Memory," *Proc. 28th Ann IEEE Symp. Foundations of Computer Science*, pp. 185–194, 1987.
[12] H.J. Siegel, *Interconnection Networks for Large-Scale Parallel Processing*. Lexington Books, 1985.
[13] C.B. Stunkel, D.G. Shea, D.G. Grice, P.H. Hochschild, and M. Tsao, "The SP1 high-performance switch," *Proc. 1994 Scalable High-Performance Computing Conf.*, pp. 150–157, 1994.
[14] L.G. Valiant and G.J. Brebner, "Universal Schemes for Parallel Communication," *Proc. 13th Ann. ACM Symp. Theory of Computing*, pp. 263–277, 1981.
[15] L.G. Valiant, "A Scheme for Fast Parallel Communication," *SIAM J. Computing*, vol. 11, no. 2, pp. 350–361, 1982.
[16] E. Upfal, "Efficient Schemes for Parallel Communication," *J. ACM*, vol. 31, pp. 507–517, 1984.
[17] Eli Upfal, "An $O(\log N)$ Deterministic Packet Routing Scheme," *21st ACM Ann. Symp. Theory of Computing*, pp. 241–250, 1989.



Eli Upfal received a BSc in mathematics in 1979, and a PhD in computer science in 1983, both from the Hebrew University, Jerusalem, Israel. During 1982-1983 he was a research fellow at the University California at Berkeley and in 1983-1984, a post-doctorate fellow at Stanford University. In 1985, he joined the IBM Almaden Research Center where he is currently a research staff member in the Foundation of Computer Science group. In 1988, he joined the Faculty of Applied Mathematics and Computer Science at the Weizmann Institute, Israel, where he is currently the Norman D. Cohen Professor of Computer Science His main research areas are theory of algorithms, randomized computing, probabilistic analysis of algorithms, communication networks, and parallel and distributed computing.

Sergio Felperin received his degree in computer science from the Escuela Superior Latino Americana de Informatica in 1990. He joined the IBM Argentina Computer Research and Advanced Applications Group from 1990 to 1993. His main research interests are parallel and distributed systems and languages, parallel communications, and systems simulation. He has taught parallel and distributed systems at the University of Buenos Aires for several years. Dr. Felperin is now with the System Architecture Group of IBM Argentina, where he develops architectures for large systems.



Marc Snir received a PhD in mathematics from the Hebrew University of Jerusalem in 1979. He worked at New York University on the NYU Ultracomputer project in 1980-1982, and worked at the Hebrew University of Jerusalem from 1982-1986. In 1986 he joined the IBM T. J. Watson Research Center. He has published on computational complexity, parallel algortihms, parallel architectures, interconnection networks, and parallel programming environments. He has recently coauthored the High Performance Fortran and the Message Passing Interface standards. Dr. Snir is a senior manager at IBM T.J. Watson Research Center, where he leads research on scalable parallel software and on scalable parallel architectures. He led the initial design and prototyping of the parallel software for the IBM SP1 and SP2, and is currently directing research on technologies for scalable parallel computing. Dr. Snir is a member of the IBM Academy of Technology, an IEEE Fellow, and a member of ACM.