An $O(\log N)$ Deterministic Packet-Routing Scheme

ELI UPFAL

IBM Almaden Research Center, San Jose, California, and The Weizmann Institute of Science, Rehovot, Israel

Abstract. A deterministic $O(\log N)$-time algorithm for the problem of routing an arbitrary permutation on an N-processor bounded-degree network with bounded buffers is presented.

Unlike all previous deterministic solutions to this problem, our routing scheme does not reduce the routing problem to sorting and does not use the sorting network of Ajtai, et al. [1]. Consequently, the constant in the run time of our routing scheme is substantially smaller, and the network topology is significantly simpler.

Categories and Subject Descriptors: C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*circuit-switching networks*; *distributed networks*; *network communications*; *network topology*; *packet networks*; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*computations on discrete structures*; *routing and layout*; G.2.2 [**Discrete Mathematics**]: Graph Theory—*network problems*

General Terms: Algorithm, Design, Performance, Theory

Additional Key Words and Phrases: Deterministic routing, network routing, parallel computer

1. *Introduction*

We present an efficient and simple deterministic solution to a fundamental problem in communication and parallel computation; the problem of routing an arbitrary permutation request on a bounded-degree network topology, with bounded buffers.

An N permutation request is a set of N packets. Each packet resides in a distinct processor and each processor is the destination of one packet. We are required to specify a bounded-degree network topology, and a routing algorithm that routes an arbitrary permutation request on that network in minimum parallel time. In our computation model, a node in the network can send and receive only one packet per step, and can store simultaneously only a bounded number of packets.

A preliminary version of this work was published in *Proceedings of the 21st Annual ACM SIGACT Symposium on Theory of Computing* (Seattle, Wash., May 15-17), ACM, New York, 1989, pp. 241-250

Eli Upfal's work at the Weizmann Institute was supported in part by a Bat-Shevu de Rothschild Award and by a Revson Career Development Award.

Author's address: IBM Research Division K53, 80L, Almaden Research Center, 650 Harry Road, San Jose, CA 95120-6099.

Packet routing can be implemented in two types of communication environments: switching networks and communication networks.

A switching network is an acyclic network with N inputs, N outputs, and at least $N \log N$ internal nodes. Each node is connected to a bounded number of input, output, and internal nodes.

A communication network has a total of N nodes, connected by a bounded degree network. Each node can generate a packet, can be the destination of a packet, and can serve as an internal node.

The problem of efficient packet routing on bounded-degree networks has been extensively studied in the past. However, almost all past research focused on probabilistic solutions. One reason is a known lower bound by Borodin and Hopcroft [5], showing that the obvious routing strategies (oblivious routing), cannot yield an efficient deterministic solution.

The only previously known $O(\log N)$ deterministic solutions reduced the routing problem to sorting and used variants of the Ajtai-Komlos-Szemeredi [1, 6, 9] sorting network to solve the sorting problem on a bounded-degree network. An $O(\log N)$-sorting network gives an $O(\log N)$ solution for the acyclic, switching network, packet-routing problem. *Leighton's Columnsort algorithm* [7] (which, in turn, is based on the AKS sorting network), gives an $O(\log N)$ packet routing on an N-node communication network. Although these solutions are asymptotically optimal, the constants involved in the run-time are big, and the network topology required for the routing is inherently complicated and nonregular.

Our new deterministic solution uses a novel scheme that does not reduce the routing problem to sorting. As a result the constant (multiplying $\log N$) in the run-time of the routing algorithm is substantially smaller. Furthermore, the network topology used by the new scheme is significantly simpler, its core consists of a constant number of butterfly networks superimposed one over the other in a certain pattern.

Although Batcher's $1/2 \log^2 N$ sorting network [4] is still the most efficient solution for any practical N, our new results suggest that exploring the difference between routing and sorting might yield more efficient deterministic routing algorithms. In particular, we developed a new technique for utilizing the special properties of expander graphs in constructing sparse routing graphs.

2. *Preliminaries*

Let $G = (A, B, E)$ be a bipartite graph, and let $\Gamma(X)$, denote the set of neighbors of a set of vertices X, that is, $\Gamma(X) = \{y \mid (x, y) \in E \text{ for some } x \in X\}$.

Definition 2.1. G is an *(α, β, n, d)-expander* if $|A| = |B| = n$, the degree of every vertex in G is d, and for every set $X \subset A$ such that $|X| \leq \alpha n$, $|\Gamma(X)| \geq \beta |X|$.

Definition 2.2. G is a depth 2 *$(\alpha, \beta, m, m/2, d)$-concentrator* if $|A| = m$, $|B| = m/2$, the degree of each vertex in A is d, the degree of each vertex in B is $2d$, and for every set $X \subset A$, such that $|X| \leq \alpha m$, $|\Gamma(X)| \geq \beta |X|$.

The basic building block of the routing network is a *splitter*.

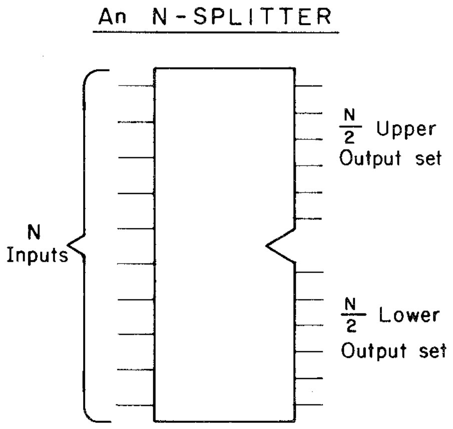

FIGURE 1

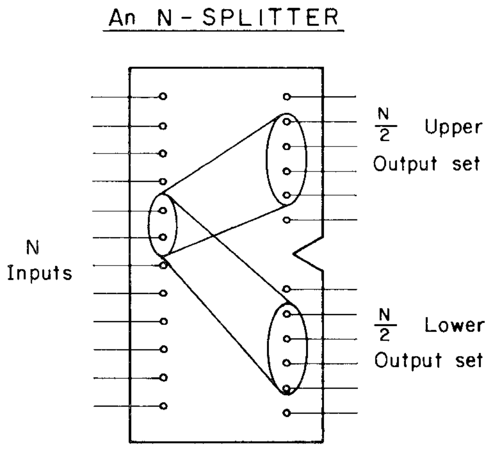

FIGURE 2

Definition 2.3. An *$(\alpha, \beta, m, 2d)$-splitter* is a depth-2 network with m inputs and two sets of $m/2$ outputs. The set of inputs is connected by an $(\alpha, \beta, m, m/2, d)$-concentrator to each of the two sets of output nodes. (See Figures 1 and 2.)

THEOREM 2.1

(1) *For any $\beta < d - 1$, there exists an (α, β, n, d)-expander with $\alpha > 1/\beta(\beta e^{1+\beta})^{-1/(d-\beta-1)}$.*

(2) *There exists an explicit construction of an (α, β, n, d)-expander for any $d = p + 1$, p prime, and $\beta \leq d/((d-4)\alpha + 4)$.*

PROOF. The first part is proven by a straightforward counting argument. The second part is a consequence of the construction given in [8]. (See [2], [3], and [10].) □

COROLLARY 2.1

(1) *For any $2\beta < d - 1$, there exists an $(\alpha, \beta, m, 2d)$-splitter with*

$$\alpha > \frac{1}{2\beta}\left(2\beta e^{1+2\beta}\right)^{-1/(d-2\beta-1)}.$$

(2) *There exists an explicit construction of an* $(\alpha, \beta, m, 2d)$*-splitter for any* $d = p + 1$, p *prime, and* $\beta \leq d/(2(d-4)\alpha + 8)$.

PROOF. An $(\alpha, \beta, m, m/2, d)$-concentrator can be constructed by folding the set of vertices B in an $(\alpha, 2\beta, m, d)$-expander $G = (A, B, E)$. Identifying the sets of A vertices in two $(\alpha, \beta, m, m/2, d)$-concentrators gives an $(\alpha, \beta, m, 2d)$-splitter. □

3. *Packet Routing on an Acyclic Switching Network*

3.1. THE MULTI-BUTTERFLY NETWORK. An (N, d, α, β)-multi-butterfly is a network with N input nodes, N output nodes, and $N(\log N - 1)$ internal nodes. (To simplify the presentation, we assume that N is a power of 2.) Nodes can store only a bounded number of packets in their buffers, and only one communication line adjacent to each node can be active at any given time. The nodes are partitioned into $\log N + 1$ stages, stage 0 to stage $\log N$. There are N nodes in each stage. The N nodes of the stage 0 are the input nodes, and the N nodes of stage $\log N$ are the output nodes. Let $\langle s, j \rangle$ denote the jth node in stage s.

The connections between the stages are constructed as follows: For each stage $0 \leq s \leq \log N$, the nodes of stage s are partitioned into 2^s sets $A_{s,0}, \ldots, A_{s,2^s-1}$. $A_{s,j} = \{\langle s, k \rangle \mid \lfloor k/2^{\log n - s} \rfloor = j\}$. The $N/2^s$ nodes of set $A_{s,j}$ are connected by an $(\alpha, \beta, N/2^s, 2\text{d})$-splitter to the two $N/2^{s+1}$ sets $A_{s+1,2j}$, and $A_{s+1,2j+1}$.[1] (See figure 3.)

Our network is strongly related to the butterfly network. The only difference between the two networks is that in the butterfly network a vertex in a set $A_{s,j}$ is connected by one edge instead of d edges to each of the sets $A_{s+1,2j}$ and $A_{s+1,2j+1}$. In Appendix A, we show that the Multi-Butterfly network can in fact be constructed by superimposing d butterfly networks.

Clearly, there are d edge disjoint paths of length $\log N$ connecting each input to each output in the Multi-Butterfly network. Furthermore, like in the butterfly network, a $\log N$ route from each input node to each output node can be easily computed by bit comparison. By our construction, the set $A_{s,j}$ is connected to the two sets $A_{s+1,2j}$ and $A_{s+1,2j+1}$; we refer to the set $A_{s+1,2j}$ as the *upper* set of outputs of the set $A_{s,j}$, and to the set $A_{s+1,2j+1}$ as the *lower* set of outputs. In transition s, a packet is sent to the upper output set if the s bit in the bit representation of its destination address is 0, and to the lower output set otherwise. Thus, if a packet passes through a node in the set $A_{s,j}$ the first s bits in the packet's destination equal the first s bits in the bit representation of j. After $\log N$ transitions, a packet reaches the singleton $A_{\log N,j}$ and the destination of the packet is j.

3.2. THE ROUTING ALGORITHM. For efficient execution of the algorithm, we partition the N packets into $L = \lceil 1/\alpha(\beta + 1) \rceil$ priority groups or batches. The batches are constructed so that no more than $\alpha m(\beta + 1)$ packets from each batch are routed through any m-splitter (we omit the parameters α, β, and d since they are fixed throughout the network). To guarantee this property,

[1]For uniformity we use splitters throughout the network. In practice, however, it would be more efficient to replace the last $\log m$ stages of the network by N/m Batcher networks.

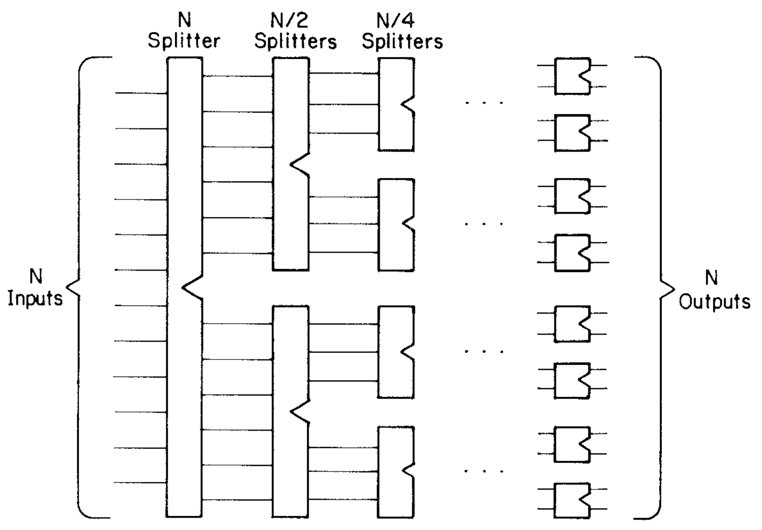

FIG. 3. Multi-Butterfly network.

batch k, $B(k)$, contains the packets with addresses in the set $\{q \mid q \bmod L = k\}$. A packet in batch $B(k)$ has higher priority than packets in $\bigcup_{j>k} B(j)$.

The edges of each splitter are colored with $2d$ colors so that no two edges of the same color are adjacent to one vertex. The algorithm works in iterations. In odd iterations, the edges connecting odd stages to even stages are activated. In even iterations, the edges connecting even stages to odd stages are activated. Edges are activated one after the other according to the color order. Thus, in each step, only one edge adjacent to each processor is activated.

Processors try to send the packet with the highest priority that they currently store. When an edge connecting v to u is activated, if the packet with the highest priority in v is from $B(k)$, and if u does not store a packet from $B(k)$, then that packet can be sent from v to u. This protocol guarantees that no more than L packets will be stored simultaneously in one processor.

3.3. THE PERFORMANCE OF THE MULTI-BUTTERFLY NETWORK. The following claims give the basic tool for analyzing the progress of packets in the network.

CLAIM 3.1. *Given an m-splitter, denote by $Y^t(j)$ the number of packets of batch j stored at input nodes of the splitter after iteration t. Denote by $Z^t(k)$ the total number of packets of batch k stored in the two output sets of the splitter after iteration t. Assume that the edges of the splitter are activated in iteration t of the algorithm. Then*

$$Y^t(k) \leq \frac{Y^{t-1}(k) + Z^{t-1}(k)}{\beta + 1} + \sum_{j=0}^{k-1} Y^{t-1}(j).$$

PROOF. In each iteration, each processor tries to send the packet with the highest priority that it currently stores. A packet in processor v is sent to a neighbor u, only if u does not store a packet from the same batch.

Let $I^{t-1}(k)$ denote the set of input nodes of the splitter storing packets of $B(k)$ before the execution of iteration t. Since a node can store only one packet

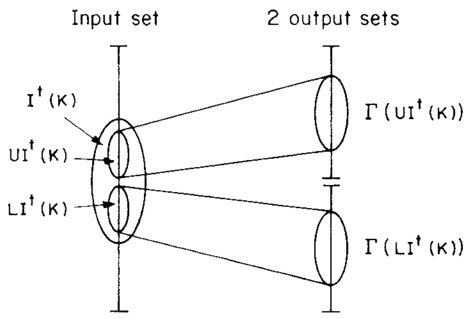

FIGURE 4

from each batch, $|I^{t-1}(k)| = Y^{t-1}(k)$. Let $HI^{t-1}(k) \subseteq I^{t-1}(k)$ denote the set of input nodes in which the packets from $B(k)$ has the highest priority, that is, processors in $I^{t-1}(k)$ that do not store packets of $\bigcup_{j<k} B(k)$ before the execution of iteration t. Let $HI^{t-1}(k) = UHI^{t-1} \cup LHI^{t-1}(k)$, where $UHI^{t-1}(k)$ stores packets that need to be sent to the upper output set, and $LHI^{t-1}(k)$ stores packets that need to be sent to the lower output set. Let $UI^t(k) \subseteq UHI^{t-1}(k)$, and $LI^t(k) \subseteq LHI^{t-1}(k)$ denote the nodes in $HI^{t-1}(k)$ that contains packets of $B(k)$ after the execution of iteration t. Note that if a node is in $UI^t(k)$ or $LI^t(k)$, all its neighbors in the upper or the lower output set respectively store packets of $B(k)$ at the end of iteration t. (See figure 4.)

The input set is connected to the upper output set by an $(\alpha, \beta, m, m/2, d)$-concentrator. If $|UI^t(k)| > \alpha m$, then by the expansion property of the concentrator

$$|\Gamma(UI^t(k))| \geq \alpha\beta m.$$

But each processor in $UI^t(k) \cup \Gamma(UI^t(k))$ stores a packet from $B(k)$, and no more than $\alpha(\beta + 1)m$ packets of $B(k)$ are routed through any splitter, thus $|UI^t(k)| \leq \alpha m$.

Since each packet in $HI^{t-1}(k)$ is either in an input or in an output node, after the execution of iteration t, we get

$$\beta|HI^t(k)| = \beta|LI^t(k)| + \beta|UI^t(k)| \leq |\Gamma(LI^t(k))| + |\Gamma(UI^t(k))|$$
$$\leq |HI^{t-1}(k)| + Z^{t-1}(k) - |HI^t(k)|.$$

Since $|HI^{t-1}(k)| \leq Y^{t-1}(k)$,

$$|HI^t(k)| \leq \frac{Y^{t-1}(k) + Z^{t-1}(k)}{\beta + 1}.$$

$Y^t(k) - |HI^t(k)| \leq \sum_{j=0}^{k-1} y^{t-1}(j)$, since each input node that stores a packet of $B(k)$ and is in $I^{t-1}(k) - HI^{t-1}(k)$ stored a packet from $\bigcup_{j<k} B(k)$ before the execution of iteration t. Thus,

$$Y^t(k) \leq |HI^t(k)| + \sum_{j=0}^{k-1} Y^{t-1}(j)$$
$$\leq \frac{Y^{t-1}(k) + Z^{t-1}(k)}{\beta + 1} + \sum_{j=0}^{k-1} Y^{t-1}(j). \qquad \square$$

Using Claim 3.1, we can prove a uniform bound on $X_s^t(\mathrm{k})$, the number of elements of batch k in column s after the execution of iteration t of the algorithm.

Fix constants α, β, γ, and θ, such that

$$\frac{4}{\beta+1} + \frac{3\gamma^\theta}{1-\gamma^\theta} \leq \gamma^2 < 1. \tag{1}$$

CLAIM 3.2. *Let t be even, then at the end of iteration t,*

$$X_s^t(k) \leq \begin{cases} \min\left[\dfrac{N}{L}, \dfrac{N}{L}\gamma^{t-\theta k}(1+\beta)^{(s+1)/2}\right] & \text{if } s \text{ is odd;} \\ \min\left[\dfrac{N}{L}, \dfrac{N}{L}\gamma^{t-\theta k}(1+\beta)^{s/2}\right] & \text{if } s \text{ is even.} \end{cases}$$

PROOF. For $t = 0$, the claim is trivially true since there are no more than N/L packets in any batch.

Assume that the claim is true for $t - 2$, and fix k. Let s be odd. In iteration $t - 1$, stage s is sending packets to stage $s + 1$.

Applying Claim 3.1 to all the splitters connecting state s to stage $s + 1$, we get

$$\begin{aligned} X_s^{t-1}(k) &\leq \frac{X_s^{t-2}(k) + X_{s+1}^{t-2}(k)}{\beta+1} + \sum_{j=0}^{k-1} X_s^{t-2}(j) \\ &\leq \frac{N}{L}\gamma^{t-2-\theta k}(\beta+1)^{(s+1)/2}\left(\frac{2}{\beta+1} + \sum_{j=0}^{k-1}\gamma^{\theta(k-j)}\right) \\ &\leq \frac{N}{L}\gamma^{t-2-\theta k}(\beta+1)^{(s+1)/2}\left(\frac{2}{\beta+1} + \frac{\gamma^\theta}{1-\gamma^\theta}\right) \end{aligned}$$

by the induction hypotheses. A similar bound holds for $X_{s+2}^{t-1}(k)$.

Since $s - 1$ is even, stage $s - 1$ receives packets from stage $s - 2$ in iteration $t - 1$. It can receive no more than $X_{s-2}^{t-1}(k)$ packets of $B(k)$ thus,

$$X_{s-1}^{t-1}(k) \leq X_{s-2}^{t-2}(k) + X_{s-1}^{t-2}(k) \leq 2\frac{N}{L}\gamma^{t-2-\theta k}(\beta+1)^{(s-1)/2}.$$

A similar bound holds for $X_{s+1}^{t-1}(k)$.

In iteration t, stage s is receiving packets from stage $s - 1$.

$$\begin{aligned} X_s^t(k) &\leq X_{s-1}^{t-1}(k) + X_s^{t-1}(k) \\ &\leq 2\frac{N}{L}\gamma^{t-2-\theta k}(\beta+1)^{(s-1)/2} \\ &\quad + \left(\frac{2}{\beta+1} + \frac{\gamma^\theta}{1-\gamma^\theta}\right)\frac{N}{L}\gamma^{t-2-\theta k}(\beta+1)^{(s+1)/2} \\ &\leq \left(\frac{4}{\beta+1} + \frac{\gamma^\theta}{1-\gamma^\theta}\right)\frac{N}{L}\gamma^{t-2-\theta k}(\beta+1)^{(s+1)/2} \\ &\leq \frac{N}{L}\gamma^{t-\theta k}(\beta+1)^{(s+1)/2} \end{aligned}$$

by the requirement (1) on γ.

Stage $s+1$ is sending packets in iteration t. Applying Claim 3.1 we get

$$\begin{aligned}
X_{s+1}^{t}(k) &\le \frac{X_{s+1}^{t-1}(k) + X_{s+2}^{t-1}(k)}{\beta+1} + \sum_{j=0}^{k-1} X_{s+1}^{t-1}(j) \\
&\le \frac{2}{\beta+1}\frac{N}{L}\gamma^{t-2-\theta k}(\beta+1)^{(s+1)/2} \\
&\quad + \frac{1}{\beta+1}\left(\frac{2}{\beta+1} + \frac{\gamma^{\theta}}{1-\gamma^{\theta}}\right)\frac{N}{L}\gamma^{t-2-\theta k}(\beta+1)^{(s+3)/2} \\
&\quad + \sum_{j=0}^{k-1} 2\frac{N}{L}\gamma^{t-2-\theta j}(\beta+1)^{(s+1)/2} \\
&\le \left(\frac{4}{\beta+1} + \frac{3\gamma^{\theta}}{1-\gamma^{\theta}}\right)\alpha N\gamma^{t-2-\theta k}(\beta+1)^{(s+1)/2} \\
&\le \alpha N\gamma^{t-\theta k}(\beta+1)^{(s+1)/2}
\end{aligned}$$

by the requirement (1) on θ. □

THEOREM 3.1. *The (N, d, α, β)-Multi-Butterfly network routes an arbitrary permutation in*

$$2d\left(\frac{1 + 1/2\,log(1+\beta)}{log\, 1/\gamma}\right) log N + O(1)$$

steps.

PROOF. We bound the total number of packets in the network after T iterations (T even), by

$$\begin{aligned}
&\sum_{k=0}^{L-1}\sum_{s=0}^{\log N-1}\frac{N}{L}\gamma^{T-\theta k}(1+\beta)^{(s+1)/2} \\
&\quad\le \frac{N}{L}(\beta+1)^{1/2(1+\log N)}\sum_{k=0}^{L-1}\gamma^{T-\theta k} \\
&\quad\le \frac{N}{L}(\beta+1)^{1/2(1+\log N)}\gamma^{T-\theta L}\frac{\gamma^{\theta}}{1-\gamma^{\theta}}.
\end{aligned}$$

This sum is less than 1 for

$$T_0 > \left(\frac{1 + 1/2\log(1+\beta)}{\log 1/\gamma}\right)\log N + O(1).$$

Each iteration takes $2d$ steps. Thus, the run-time is bounded by $2dT_0 + O(1)$. □

A reasonable choice of parameters for a nonexplicit construction $d = 10$, $\beta = 7$, $\alpha = 1/2312$, $\gamma = 0.714$, $\theta = 14$, and $L = 68$. The run-time for that assignment is $100\log N + O(1)$ compared to $6100\log N + O(1)$ using an AKS sorting network [9].

A reasonable choice for an explicit construction is $d = 60$, $\beta = 7$, $\alpha = 1/189$, $\gamma = 0.707$, $\theta = 14$, and $L = 27$, which yields run-time $600 \log N + O(1)$ compared to at least $10^6 \log N + O(1)$ using an AKS sorting network.

Note that the Multi-Butterfly solution uses significantly less hardware. The Multi-Butterfly network has only $\log N$ stages each with N nodes. The number of stages in the AKS network is equal to its run-time, each stage with N nodes. Although the degree of each node and the size of the node's buffer is larger in our network, the total number of edges in the network, and the total number of buffers is smaller than in the sorting network. By replacing each set of d edges adjacent to one node with a tree of depth $\log_3 d$, one can decrease the degree to 4 adding no more than a factor of $2 \log_3 d$ to the run-time. Thus, we get a nonexplicit construction of degree 4 that runs in $600 \log N$ steps and an explicit construction of degree 4 that runs in $4800 \log N$ steps.

4. *Routing N Packets on an N-Processor Communication Network*

4.1. The Network Topology. To simplify the presentation, we use topology with $3N$ nodes. We later show how to implement the topology on an N-processor network. Nodes can store only a bounded number of packets in their buffers. Only one communication line adjacent to each node is active at any given time. When a communication line is active, the nodes adjacent to it can either exchange the content of one of their buffers or keep their buffers unchanged.

Let $N = n(\log n + 1)$. The nodes are partitioned into $3 \log n + 1$ stages, stage 0 to stage $3 \log n$. Each stage contains n nodes, and for every $0 < s \leq 3 \log n - 1$, stage s is connected to stage $s - 1$ and stage $s + 1$. A node in this topology has a unique address $\langle s, q \rangle$, where $0 \leq s \leq 3 \log n$ denote the stage in which the node is located, and $1 \leq q \leq n$ is its number in the stage.

The first $\log n + 1$ stages are connected by expanders. For each $0 \leq s < \log n$, stage s connected by an (α, β, n, d)-expander to stage $s + 1$. The intermediate $\log n + 1$ stages are connected by an (N, d, α, β)-Multi-Butterfly network. The n nodes of stage $\log n$ are connected by an n-splitter to two sets of $n/2$ nodes of stage $\log n + 1$. For each $0 \leq i \leq \log n - 1$ and $1 \leq j \leq 2^i$, the set $A_{i,j}$ of stage $\log n + i$ is connected by an $n/2^i$*-splitter* to the sets $A_{i+1,2j-1}$ and $A_{i+1,2j}$ of stage $\log n + i + 1$.

The last $\log n + 1$ stages are connected to each other only by forward edges. Each node $\langle s, q \rangle$, $2 \log n \leq s \leq 3 \log n$ is connected to node $\langle s + 1, q \rangle$.

Initially all the N packets reside in the first $\log n + 1$ stages. All the final destinations of the packets are in the nodes of the last $\log n + 1$ stages. Clearly, there is a path with no more than $3 \log n$ edges between every node in the first $\log n + 1$ stages and every node in the last $\log n + 1$ stages, and this path can be locally computed. A packet initially at node $\langle s, q \rangle$ with destination $\langle s', q' \rangle$ can take an arbitrary path forward to stage $\log n$. By bit comparison, the packet is then led to the node $\langle 2 \log n, q' \rangle$, and then by the direct edges $(\langle x, q' \rangle, \langle x + 1, q' \rangle)$, the packet reaches its destination.

4.2. The Routing Algorithm. The core of the routing is the Multi-Butterfly network. Given a packet with destination $\langle s, q \rangle$, the Multi-Butterfly network delivers the packet to node $\langle 2 \log n, q \rangle$. From there the packet proceeds through the direct edges to its destination. To achieve an $O(\log N)$

run-time we pipeline $O(\log n)$ batches, each with $O(n)$ packets. The main difficulty in implementing this idea is that the Multi-Butterfly network can handle efficiently only partial permutations on n, while an arbitrary set of $O(n)$ packets might contain up to $\log n$ packets that need to reach the same output of the Multi-Butterfly network. The task of the first $\log n$ stages is to feed the Multi-Butterfly network with well-prepared batches, that is, to approximately sort the packets according to the column of their destination.

For efficient execution, we partition the N packets into $\lceil 1/\alpha \rceil(\log n + 1)$ batches. For $k = 0, \dots, \lceil 1/\alpha \rceil(\log n + 1) - 1$, batch k, $B(k)$, consists of no more than αn packets with destinations in the set $\{\langle s, q\rangle \mid s = \lfloor k/\lceil 1/\alpha \rceil \rfloor$, and $q = k \bmod \lceil 1/\alpha \rceil\}$. This construction guarantees that no more than αm packets from each batch can reach the input set of any m-splitter. A packet in $B(k)$ has higher priority over packets in $\bigcup_{j>k} B(j)$.

The edges of each expander are colored by d colors, and the edges of each splitter are colored by $2d$ colors, such that no two edges with the same color share a node. The algorithm works in iterations. In odd iterations, the edges connecting odd stages to even stages are activated while in even iterations edges connecting even stages to odd stages are activated.

In each iteration, the edges are activated one after the other according to the color order. When an edge-connecting node $\langle t, q\rangle$ to node $\langle t+1, q'\rangle$ is activated, if node $\langle t, q\rangle$ stores in its buffer a packet with higher priority than the packet stored in the buffer of $\langle t+1, q'\rangle$, the two nodes exchange packets. An empty buffer is considered a packet with the lowest priority.

An iteration takes d steps of a node in the first $\log n$ stages, $2d$ steps of a node at the second $\log n$ stages, and 1 step of a node in the third $\log n$ stages. Thus, we can simulate the network on N processors. Each processor has degree $6d + 2$, and 3 buffers. An execution of each iteration, by the N processors, takes $3d + 1$ steps.

In Section 4.4, we show how to reduce the degree of the network without significantly changing its performance.

4.3. Analysis of the Network Performance. Let $X_s^t(K)$ denote the number of elements of batch k in column s after the execution of iteration t of the algorithm.

Claim 4.1. *Let $s < \log n$, and let t be an iteration in which stage s is sending packets to stage $s + 1$. Then*

$$X_s^t(k) \le \frac{\sum_{j=0}^{k}\left(X_s^{t-1}(j) + X_{s+1}^{t-1}(j)\right)}{\beta + 1}.$$

Proof. Since $s < \log n$, stage s is connected to stage $s+1$ by an (α, β, n, d)-expander. Let X denote the locations of the $X_s^t(k)$ packets of batch k that were left in stage s after the execution of step t. Since packets in batch $B(k)$ have priority over packets in $\bigcup_{j>k} B(j)$ all neighbors of X in stage $s+1$ contain packets of $\bigcup_{j \le k} B(j)$.

Since batch k has no more than αn packets, $|X| = X_s^t(k) \le \alpha n$. By the expansion property

$$\beta |X| \le |\Gamma(X)| \le \sum_{j=0}^{k} \left(X_s^{t-1}(j) + X_{s+1}^{t-1}(j)\right) - |X|.$$

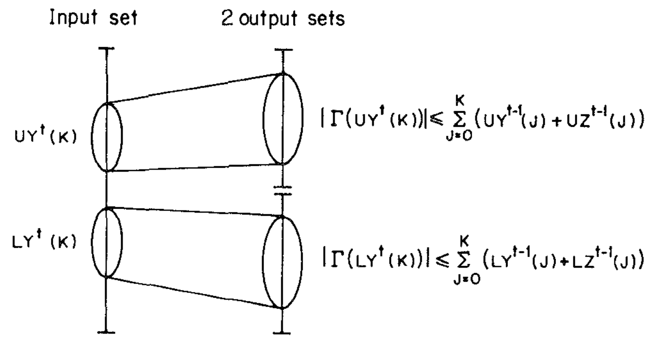
Input set
2 output sets
UY^t (K)
LY^t (K)
|Γ(UY^t(K))| ≤ Σ_{J=0}^{K} (UY^{t-1}(J) + UZ^{t-1}(J))
|Γ(LY^t(K))| ≤ Σ_{J=0}^{K} (LY^{t-1}(J) + LZ^{t-1}(J))

FIGURE 5

Thus

$$X_s^t(k) = |X| \le \frac{\sum_{j=0}^{k}\left(X_s^{t-1}(j) + X_{s+1}^{t-1}(j)\right)}{\beta + 1}. \qquad \square$$

CLAIM 4.2. *Given an m-splitter, denote by $Y^t(j)$ the number of packets of batch j stored at input nodes of the splitter after iteration t. Denote by $Z^t(j)$ the total number of packets of batch j stored in the two output sets of the splitter after iteration t. Assume that the edges of the splitter are activated in iteration t of the algorithm, then*

$$Y^t(k) \le \frac{\sum_{j=0}^{k}\left(Y^{t-1}(j) + Z^{t-1}(j)\right)}{\beta + 1}.$$

PROOF. Let $Y^t(j) = UY^t(j) + LY^t(j)$, where $UY^t(j)$ packets need to be sent to the upper output set and $LY^t(j)$ to the lower output set. Let $Z^t(j) = UZ^t(j) + LZ^t(j)$ where $UZ^t(j)$ packets are in the upper output set and $LZ^t(j)$ are in the lower output set after iteration t. (See Figure 5.)

The input set is connected to the upper output set by an $(\alpha, \beta, m, m/2, d)$-concentrator. Let X denote the locations of the $UY^t(k)$ packets of batch k that were left in input nodes of the splitter after the execution of iteration t. (There is no more than one packet of $UY^t(k)$ in each node.) Since packets in batch $B(k)$ have priority over packets in $\bigcup_{j>k} B(j)$, all neighbors of X in the upper output set contain packets of $\bigcup_{j\le k} B(j)$.

Since no more than αm packets of batch k can reach the m-splitter, $|X| = UY^t(k) \le \alpha m$. By the expansion property of the concentrator

$$\beta |X| \le |\Gamma(X)| \le \sum_{j=0}^{k} \left(UY^{t-1}(j) + UZ^{t-1}(j)\right) - |X|.$$

Thus,

$$UY^t(k) = |X| \le \frac{\sum_{j=0}^{k}\left(UY^{t-1}(j) + UZ^{t-1}(j)\right)}{\beta + 1}.$$

A similar inequality holds for $LY^t(k)$. Combining both inequalities, we get

$$Y^t(k) \le \frac{\sum_{j=0}^{k}\left(Y^{t-1}(j) + Z^{t-1}(j)\right)}{\beta + 1}. \qquad \square$$

Using Claims 4.1 and 4.2, we can prove a uniform bound on the number of packets of a given batch stored in a given stage after the execution of iteration t.

Fix constants α, β, γ, and θ satisfying the following condition:

$$\frac{2}{\gamma^2(\beta+1)(1-\gamma^\theta)}\left(1+\frac{1}{1-\gamma^\theta}\right)\le 1. \tag{2}$$

Claim 4.3. *Let t be even; then*

$$X_s^t(k) \le \begin{cases} \min\left[\alpha n,\ \alpha n\gamma^{t-\theta k}(1+\beta)^{(s+1)/2}\right] & \text{if } s \text{ is odd;}\\ \min\left[\alpha n,\ \alpha n\gamma^{t-\theta k}(1+\beta)^{s/2}\right] & \text{if } s \text{ is even.}\end{cases}$$

Proof. We prove by induction on t. For $t = 0$, the claim is trivially true since all bounds equal the maximum number of packets of batch k that can reach that set.

Assume that the claim is true for $t - 2$, and fix k. Let s be odd. If $s < \log n$, we apply Claim 4.1, else we apply Claim 4.2.

Since s is odd, stage s is sending packets in iteration $t - 1$. By the induction hypothesis and Claim 4.1 or 4.2:

$$\begin{aligned} X_s^{t-1}(k) &\le \frac{\sum_{j=0}^{k}\left(X_s^{t-2}(j)+X_{s+1}^{t-2}(j)\right)}{\beta+1}\\ &\le 2\alpha n\sum_{j=0}^{k}\frac{\gamma^{t-2-\theta j}(1+\beta)^{(s+1)/2}}{(\beta+1)}\\ &\le 2\alpha n\frac{\gamma^{t-2-\theta k}(1+\beta)^{(s+1)/2}}{(\beta+1)(1-\gamma^\theta)}. \end{aligned}$$

Since $s - 1$ is even, in iteration $t - 1$ it receives packets from stage $s - 2$. It can receive no more than $X_{s-2}^{t-2}(k)$ packets of batch k. Thus,

$$X_{s-1}^{t-1}(k) \le X_{s-2}^{t-2}(k) + X_{s-1}^{t-2}(k) \le 2\alpha n\gamma^{t-2-\theta k}(1+\beta)^{(s-1)/2}.$$

In the next iteration (iteration t), stage s receives packets from stage $s - 1$.

$$\begin{aligned} X_s^t(k) &\le X_{s-1}^{t-1}(k) + X_s^{t-1}(k)\\ &\le 2\alpha n\gamma^{t-2-\theta k}(1+\beta)^{(s-1)/2}\\ &\quad + 2\alpha n\frac{\gamma^{t-2-\theta k}(1+\beta)^{(s+1)/2}}{(\beta+1)(1-\gamma^\theta)}\\ &\le \alpha n\gamma^{t-\theta k}(1+\beta)^{(s+1)/2} \end{aligned}$$

by (2).

Stage $s + 1$ is sending packets in iteration t.
Using Claim 4.1 or 4.2, and condition (2), we get

$$\begin{aligned} X^t_{s+1}(k) &\le \frac{1}{\beta+1}\sum_{j=0}^{k} 2\alpha n \gamma^{t-2-\theta j}(\beta+1)^{(s+1)/2} \\ &\quad + \frac{1}{\beta+1}\sum_{j=0}^{k} 2\alpha n \frac{\gamma^{t-2-\theta j}(1+\beta)^{(s+3)/2}}{(\beta+1)(1-\gamma^{\theta})} \\ &\le \alpha n \gamma^{t-\theta k}(1+\beta)^{(s+1)/2}. \end{aligned}$$ □

THEOREM 4.1. *The network routes an arbitrary N-permutation in*

$$(3d+1)\left(\frac{\theta}{\alpha} + \frac{1 + log(1+\beta)}{log(1/\gamma)} + 1\right) + O(1) log N$$

steps.

PROOF. We bound the total number of packets that are in the first $2\log n$ stages of network after T iterations (T even). There are $K_0 = (n \log n)/\alpha n$ batches. Thus, the total number of packets in the first $2 \log n$ stages is bounded by

$$\sum_{j=0}^{K_0-1}\sum_{s=0}^{2\log n-1} \alpha n \gamma^{T-\theta j}(\beta+1)^{(s+1)/2} \le \alpha n(\beta+1)^{\log n+1}\gamma^{T-\theta K_0}\frac{\gamma^{\theta}}{1-\gamma^{\theta}}.$$

This sum is less than 1 for

$$T_0 \ge \left[\frac{\theta}{\alpha} + \frac{1+\log(1+\beta)}{\log(1/\gamma)}\right]\log n + O(1).$$

After an additional $\log n$ iterations all packets reach their final destinations through the direct edges of the last $\log n$ stages. An iteration takes $3d + 1$ steps; thus, the total time is bounded by $(3d + 1)(T_0 + \log n)$. □

For the nonexplicit case, fix $d = 21$, $\beta = 6$, $\gamma = 0.81$, $\alpha = 0.042$, and $\theta = 12$. This assignment yields a run-time of $18\,500 \log N$ steps.

For the explicit case, fix $d = 102$, $\beta = 6.4$, $\gamma = 0.796$, $\alpha = 0.04$, and $\theta = 10$. This yields a run-time of $8 \times 10^4 \log N$ steps.

4.4. REDUCING THE NETWORK DEGREE. We use the repetitive structure of the network to reduce its degree to 10, adding no more than a factor of 2 to the run-time of the routing algorithm. We assume for that reduction that d is a power of 2.

THEOREM 4.2. *Let $G = (V, U, E)$ be a bi-partite graph specifying the connections between two adjacent stages in the network. Assume that*

(1) *The degree of G is d.*
(2) *Each node stores no more than one packet in each step.*
(3) *There is a partition of G into d identical graphs $H_i = (A_i, B_i, E(H_i))$, $i = 0, \dots, d-1$ such that $V = \bigcup_{j=0}^{d-1} A_i$, $U = \bigcup_{j=0}^{d-1} B_i$, and $E = \bigcup_{j=0}^{d-1} E(H_i)$.*

Then, there is a network $G' = (V \cup U, E')$, such that the degree of G' is 3, *and each step of the original stage can be simulated on G' in* 2 *steps.*

Proof. Let $H = (A, B, E(H))$ denote the generic graph of H_i, $i = 0, \dots, d-1$. For each vertex v in H, denote by $v_0, \dots, v_{d-1}$ the d copies of v in the d graph $G = \bigcup_{j=0}^{k-1} H_i$.

To construct the graph G', we color the edges of H by d colors such that no vertex is adjacent to two edges with the same color. The graph G' has the same set of vertices as the graph G. The edges are constructed as follows: For each edge (v, u) in H that got the color i we connect v_i to u_i in G'. For each vertex v in H, we connect the set of vertices $v_0, \dots, v_{d-1}$ with a cycle: v_i is connected to $v_{(i+1) \bmod d}$.

In each iteration of the original network, each processor activates the edges adjacent to it, one after the other according to the color order. In the new implementation, each processor in the original network is a virtual process that moves along a cycle of d nodes. In each node, the virtual process can communicate with one of processes adjacent to it in the original network.

We simulate an iteration of the original network by d phases of the new network. Denote by $\tilde{v}_i$ the virtual process of vertex v_i in the original network $G = \bigcup_{j=0}^{d-1} H_i$. The set of processes $\tilde{v}_0, \dots, \tilde{v}_{d-1}$ is simulated by the cycle of the vertices $v_0, \dots, v_{d-1}$. In each phase, each processor first communicates with another process, then moves to the next vertex in its cycle.

In phase k, process $\tilde{v}_i$ is in vertex $v_{(i+k) \bmod d}$. Let u be the neighbor of v in H, such that the edge (v, u) was colored by $(i+1) \bmod d$. By the construction of the network, vertex $v_{(i+k) \bmod d}$ is connected to vertex $u_{(i+k) \bmod d}$. That vertex contains, in phase k, the process u_i. Since v is connected to u in H, v_i is connected to u_i in G, and the processes $\tilde{v}_i$ and $\tilde{u}_i$ can simulate step $(i+k) \bmod d$ of their iteration in the original algorithm. Thus, in d phases each process can communicate with the d processes adjacent to it in the original network.

To move a process to the next vertex in the cycle, only the content of the buffer associate with this process needs to be sent. Thus, each phase takes two steps. One step to communicate between the processes and one step to forward them to the next vertex in their cycle. □

Theorem 4.3. *The construction of Section* 4.1 *can be implemented on an N-node network with degree* 10, *such that the run-time of the routing algorithm on the new network is bounded by*

$$(6d+1)\left(\frac{\theta}{\alpha} + \frac{1 + \log(1+\beta)}{\log(1/\gamma)} + 1\right)\log N$$

steps.

Proof. We first construct a network of $3N$ processors and then simulate it on an N-processors network. Let $N = n(\log n + d \log d)$. We start with a network that consists of $3 \log n + 4d \log 2d + 1$ stages, each stage with n processors. The first $\log n + 2d \log 2d + 1$ stages are connected to each other by identical (α, β, n, d)-expanders. The next $\log n + 1$ stages perform a Multi-Butterfly network, and the last $\log n + 2d \log 2d + 1$ stages are connected only by direct edges.

We can apply Theorem 4.2 to sets of d nonadjacent expanders to reduce the degree of all the vertices in the first part of the network to 4. (We assume that d divides $1/2 \log n + 2d \log 2d + 1$; otherwise, we treat the nodes in the remaining stages in the same way we treat the nodes of the first $\log 2d$ stages of the Multi-Butterfly.)

We replace each node in the first $\log 2d$ of the Multi-Butterfly by a cycle of $2d$ vertices. Each new vertex is connected to one backward and one forward edge of the original node. Thus, we add $2nd \log 2d$ new vertices each with degree 4.

We apply Theorem 4.2 to the remaining stages of the Multi-Butterfly network and reduce the degree of all the vertices there to 4. (Since we assumed that d is a factor of 2, once a stage consists of more than d splitters, it can be partitioned into d identical isolated patterns.)

We get a graph with $3N$ processors, $2N$ processors have degree 4, and N processors have degree 2. Each iteration takes $2d$ steps in the first $\log n + 2d \log 2d$ stages, $4d$ steps in the Multi-Butterfly part, and one step in the last part of the network.

Thus, we can simulate each iteration on N processors with degree 10 in $6d + 1$ steps. Since the $O(\log N)$ batches are pipelined through the network, the extra $1/\alpha 2d \log 2d$ batches of packets add only a constant to the run-time of the algorithm (see proof of Theorem 4.1). □

Using the parameters assignment of the previous section, we get a nonexplicit construction that routes a permutation in $37\,000 \log N$ steps and an explicit construction that requires $1.6 \times 10^5 \log N$ steps.

Appendix A. Constructing a Multi-Butterfly from Butterfly Networks

We show how an $N \log N$ node Multi-Butterfly network with degree d can be constructed by superimposing d butterfly networks, each with $N \log N$ nodes.

Let $B_1, \ldots, B_d$ denote d, $N \log N$ node, butterfly networks. For $s = 0, \ldots, \log N - 1$ let G_s denote an $(\alpha, \beta, N/2^{s+1}, d)$-expander. Assume that the edges of each G_i are colored by d colors and let $\pi_1^s, \ldots, \pi_d^s$ denote the d permutations defined by the d colors of the edges of G_s.

Let $\langle i, s, k\rangle$ denote the kth node in stage s of B_i, and let $A_{s,j}^i = \langle i, s, k\rangle \,|\, \lfloor k/2^{\log n - s} \rfloor = j\}$. By the construction of the butterfly, each node in $A_{s,j}^i$ is connected to one node in $A_{s+1,2j}^i$ and one node in $A_{s+1,2j+1}^i$. Furthermore, for each $0 \le q \le 2^{s-1}$, the nodes $\langle i, s, j2^s + q\rangle$, and $\langle i, s, j2^s + 2^{s-1} + q\rangle$ are connected to the same two nodes in stage $s + 1$.

We label the nodes of the butterfly networks in the following way: The label of a node $\langle i, 0, k\rangle$ is k. For $s > 0$, if the label of $\langle i, s-1, k\rangle$ is r, and $r \le 2^s$, then the label of the two nodes connecting to it in stage $s+1$ is $\pi_i^s(r)$. Note that these two nodes are also connected to $\langle i, s, k + 2^s\rangle$.

To construct the Multi-Butterfly network, we identify nodes with identical labels. The set $A_{s,j}$ of the Multi-Butterfly network is constructed by identifying nodes with identical labels in the set $\bigcup_{i=1}^d A_{s,j}^i$. Since for each set s, and i, π_i^s is a permutation, each label appears exactly d times in $\bigcup_{i=1}^d A_{s,j}^i$. Thus, the network has degree $2d$.

For each s and j, let

$$A_{s,j}^1 = \left\{\langle s, k\rangle 1 \left\lfloor \frac{k}{2^{\log n - s - 1}} \right\rfloor = j\right\}$$

and let

$$A^2_{s,j} = \left\{ \langle s, k \rangle 1 \left\lfloor \frac{k}{2^{\log n - s - 1}} \right\rfloor = 2^s + j \right\}.$$

$A_{s,j} = A^1_{s,j} + A^2_{s,j}$ and the edges define four $(\alpha, \beta, N/2^{s+1}, d)$-expanders connecting $A^1_{s,j}$ to $A_{s+1,2j}$, $A^1_{s,j}$ to $A_{s+1,2j+1}$, $A^2_{s,j}$ to $A_{s+1,2j}$, and $A^1_{s,j}$ to $A_{s+1,2j+1}$. Thus, there is an $(\alpha, \beta/2, N/s^2, N/2^{s+1}, 2d)$-splitter between each set $A_{s,j}$ and the sets $A_{s+1,2j}$ and $A_{s+1,2j+1}$, and the construction gives a Multi-Butterfly network.

ACKNOWLEDGMENT. I wish to thank Miki Ajtai, Noga Alon, Nick Pippenger, and Eli Shamir for many helpful discussions, David Peleg, and the two referees for their comments on earlier versions of this paper, and Ron Fagin for suggesting the name Multi-Butterfly.

REFERENCES

1. AJTAI, M., KOMLOS, J., AND SZEMEREDI, E. An $O(n \log n)$ sorting network. In *Proceedings of the 15th Annual ACM Symp. on Theory of Computing*. ACM, New York, 1983, pp. 1-9.
2. ALON, N. Eigenvalues and expanders. *Combinatorica 6*, 2 (1986), 83-96.
3. ALON, N. Expanders, sorting in rounds and superconcentrators of limited depth. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing* (Providence, R.I., May 6-8). ACM, New York, 1985, pp. 98-62.
4 BATCHER, K. Sorting networks and their applications. In *Proceedings of the AFIPS Spring Joint Conference*, vol. 32. AFIPS Press, Reston, Va., 1988, pp. 307-314.
5. BORODIN, A., AND HOPCROFT, J. E. Routing, merging, and sorting on parallel models of computing. *J. Comput. Syst. Sci. 30* (1985), 130-145
6. COLE, R., AND DUNLAING, C. O. Note on the AKS Sorting Network. Ultracomputer Note #109. Computer Science Department Tech. Rep. #243. New York University, New York, 1988.
7 LEIGHTON, T. Tight bounds on the complexity of parallel sorting. In *Proceedings of the 16th Annual ACM Symposium on Theory of Computing*. ACM, New York, 1984, pp. 71-80.
8. LUBOTZKY, A., PHILLIPS, R., AND SARNAK, P. Ramanugan conjecture and explicit constructions of expanders. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing*. ACM, New York, 1986, pp. 240-246.
9. PATERSON, M. S. Improved sorting network with $O(\log N)$ depth. Res. Rep. #89. Dept. of Computer Science. Univ. Warwick, Warwick, R I., 1987.
10. TANNER, R. M. Explicit concentrators from generalized N-gons. *SIAM J. Algeb. Disc. Meth. 5* (1984), 287-293.

RECEIVED FEBRUARY 1989; REVISED AUGUST 1990, ACCEPTED SEPTEMBER 1990