STOC(Milwaukee 1981), 263-277.

UNIVERSAL SCHEMES FOR PARALLEL COMMUNICATION

L.G. Valiant and G.J. Brebner,
Computer Science Department,
Edinburgh University,
Edinburgh, Scotland.

1. Introduction

Although parallel computation has been studied intensively for more than a decade the following central question has remained tantalizingly open: Is it feasible to build a general purpose computer that can exploit in arbitrary problems the inherent parallelism present in them. In this paper we isolate a combinatorial problem that, we believe, lies at the heart of this question and provide some encouragingly positive solutions to it. We show that there exists an N-processor realistic computer that can simulate arbitrary idealistic N-processor parallel computations with only a factor of $O(\log N)$ loss of runtime efficiency. The main innovation is an $O(\log N)$ time randomized routing algorithm. Previous approaches were based on sorting or permutation networks, and implied loss factors of order at least $(\log N)^2$.

Several models of parallel computation have been suggested in the literature. In the context of efficient synchronous multiprocessor algorithms there is just one model that is widely used. This consists essentially of N processors, each with general purpose sequential capabilities, that share a common memory and can access it in parallel almost arbitrarily. The access restriction, which is guaranteed during algorithm design, is that there are no store conflicts (i.e. no storage area is written to simultaneously by more than one, or some other constant number, of processors). Optionally, also, fetch conflicts are prohibited. This shared memory model reflects our intuitions about intrinsic parallelism quite well. Unfortunately no direct realization of it appears feasible in foreseeable technologies. For this reason we shall call this the idealistic model. It is widely used implicitly in the literature. One attempt to formalize it can be found in [7].

By a realistic parallel computer we mean a large number N of processors, connected together by directed edges with at most d edges entering and at most d leaving each processor. Physical limitations dictate in practice that d should be small. We incorporate this in the model by assuming throughout that d is either a constant independent of N, or a slowly growing function such as $\log_2 N$. Each processor will be a universal sequential computer in the customary sense with some local memory.

The fundamental problem that arises in simulating on a realistic machine one step of an idealistic computation is that of simulating arbitrary connection patterns among the processors via a fixed sparse network. This can be formulated within packet-switching terminology: Our proposed general purpose parallel computer U has some packets of information at each node. Each packet has a destination address written on it that specifies the node to which it is to be sent. The task of U is to route the packets to their correct destinations simultaneously so that at most one packet passes down any wire at any time and all of them arrive at their destination quickly, i.e. within $O(\log N)$ time.

The above restrictions impose some limits on

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

©1981 ACM 0-89791-041-9 /80/0500/0263 $00.75

the communication patterns that can be simulated fast. For example, if initially there are h packets at some node then it will need at least h/d time units merely to transport these away from the node. Hence to achieve runtime $\log N$ we certainly cannot place more than $d.\log N$ packets at any node initially. An attractive paradigmatic subproblem is therefore that of realising permutations (i.e. initially one packet at every node, all with distinct destination addresses.) For practicality it is necessary to be able to realise also partial permutations (i.e. initially at most one packet at every node, all with distinct destination addresses.) In fact our techniques will solve directly the more general case of partial h-relations (i.e. initially at most h packets at any node, with no destination occurring on more than h packets.)

In our formulation each packet is atomic but carries with it some tickets that contain all the book-keeping information necessary (e.g. destination address, routing information, scratch pads.) From the considerations of the previous paragraph it should be clear that within time $\log N$ information from at most $d.\log N$ packets can be gathered to any one node from other nodes. It follows that for routing the packets the strategy will have to be based on only a minute fraction of the total information necessary to specify the complete communication pattern. We conclude that the routing algorithm has to be highly distributed. (The same conclusion follows more fundamentally from a simple information theoretic argument and therefore applies also in the case when book-keeping information can be communicated on its own, and not only when accompanying packets.) The reader should note that otherwise simple questions may become problematic in a distributed context. For example, realising a partial permutation is no longer trivially reducible to the problem of realising a permutation.

The problem of realising permutations in sparse networks has been studied in a non-distributed context for some time. Permutation networks [3] achieve delay $2\log_2 N$ and can be implemented on an N-node constant-degree graph namely the shuffle/exchange [14]. Unfortunately it is difficult to parallelize this. Even on the idealistic parallel model the best bound known for computing the routing is $O((\log N)^2)$ [7,8], while on realistic models it is $O((\log N)^4)$ [5,8,11]. An alternative approach is to use Batcher's sorting networks [2]. Although these are distributed and also implementable on sparse graphs [9] the best bound is again $O((\log N)^2)$, and adaptation to partial permutations involves further losses by constant factors.

The purpose of this paper is to present schemes that can realise arbitrary permutations (or even partial h-permutations for constant h) in time $O(\log N)$. The algorithms use randomization [10,13]. They are always correct, and terminate in the required time with overwhelming probability. They have the additional property called testability, of having the same probabilistic behaviour for all permutation requests. Hence its run time and local storage requirements can be determined with accuracy by running enough Monte-Carlo simulations on any one fixed permutation.

The ideas in this paper originate from an earlier algorithm [16] for which some experimental results were reported in [17]. There $O(\log N)$ runtime was provably achieved for the Boolean n-cube (which has degree $\log N$.) Here we shall first give an alternative algorithm for the cube that has a simpler and more general proof. We then give schemes based on shuffle graphs that, on experimental evidence, achieve runtime $O(\log N)$ with only constant degree, and provably achieve $o(\log N)$ time with $o(\log N)$ degree. Finally in Section 8 we give a (nontestable) routing scheme for k-dimensional grid graphs that, essentially, run in time $(2k-1)N^{1/k}$ and require local storage for $O(\log N)$ packets at each node. This should be compared with the implementations of Batcher's sorters due to Thomson and Kung [15] that can route full permutations in time $6N^{1/2}$ if $k = 2$ and $(3k^2+k)N^{1/k}$ for $k > 2$.

The results as stated assume that the transmission times for all connections are equal and that they dominate all other steps. If this condition holds then the schemes based on the d-way shuffle (Section 6) appear to be ideal. If transmission times are proportional to physical distance the above assumption is unrealisable unfortunately and it becomes logical to use grids for interconnections. The schemes of Section 8 then become relevant.

A good survey of general purpose interconnection patterns is given by Siegel [12]. For very recent work on general purpose parallel computers see Schwartz [11], and Galil and Paul [5].

2. A General Class of Parallel Communication Schemes

The schemes considered in this paper all conform to the following pattern of parallel communication schemes (PCS):

(a) There is a directed graph $G=(V,E)$ where V is a set of N nodes $\{0,1,\ldots,N-1\}$. The nodes all have indegree d and outdegree d. Hence the number of edges is $|E| = dN$.

(b) There are a total of T atomic packets in the system at any time. Each carries with it some tickets for book-keeping. Initially the tickets contain no information except for the name of the packet and its destination address. During execution the tickets may be altered. The routing of a packet depends, however, only on the current contents of its ticket. Such a scheme is said to be oblivious (or non-adaptive) if for every packet alterations in its ticket never depend on the presence, contents or tickets of any other packet. (Typically alterations merely record the progress made towards the destination.)

(c) At every integral instant (i.e. $t=0,1,2\ldots.$) each packet is at some node. During each unit interval (i.e. time period $(t,t+1)$ for some integral t) each edge can transmit one packet in the sense of its direction.

(d) At every node there are d queues each one associated with one of the edges directed out of the node. At the start of each unit interval each queue contains a set of packets and the queuing discipline determines which one is to be at the head of the queue provided the latter is nonempty. During the unit interval the head of each nonempty queue is transmitted along the associated directed edge to the neighbouring node. Our results hold for every queuing discipline that has the minimal property of defining a head for every queue that is nonempty.

(e) At the end of each unit interval the packets at a node may consist of those that have just arrived from a neighbouring node, others that have been waiting in one of the queues during this last interval, and others still that were already finished (i.e. had arrived at their final destinations) before this interval. The routing algorithm decides for each unfinished packet on the basis of its ticket which queue it is to be in at that node.

An initialised scheme is a pair (S,IC) where S is a PCS of the kind described in (a)-(e) above, and IC describes initial conditions for it. IC specifies how the packets and their destinations are distributed at time zero.

Definition If e is an edge in (S,IC) then $Traff(e)$ the traffic in e, is the expected number of distinct packets that ever pass along e in a run of (S,IC). [N.B. Probabilities may enter both through probabilistic assumptions in IC and through randomization in the algorithm.]

Definition An initialised scheme (S,IC) is symmetric iff for any two edges e_1,e_2 in S, $Traff(e_1) = Traff(e_2)$.

The delay of a packet X in a run of (S,IC) is the total number of time units during which X waits unserved in queues. Its route is the path taken in G. [N.B. The time at which the packet X becomes finished is clearly the sum of its route length and its delay.] A scheme is nonrepeating if whenever two packets take paths $e_1e_2\ldots e_r$ and $\bar{e}_1\bar{e}_2\ldots\bar{e}_s$ in which $e_i=\bar{e}_j$ and $e_k=\bar{e}_m$ $(k>i)$ it is the case that $k-i = m-j$ and for all p $(i \le p \le k)$ $e_p = \bar{e}_{p+j-i}$.

3. Preliminary Facts

We first prove a lemma that is sufficient to guarantee that the later results hold for arbitrary queuing disciplines. To state it we define a queue line to be a directed path of length n with nodes $x_1,\ldots,x_{n+1}$, edges $\{e_i = (x_i,x_{i+1}) \mid 1 \le i \le n\}$ and a queue q_i at each node $x_i (1 \le i \le n)$, together with a set of triples $\{(i_j, t_j, r_j) \mid 1 \le j \le m\}$ that specifies that there are m packets $X_1,\ldots,X_m$, and that for each $j (1 \le j \le m)$ X_j is added to q_{i_j} at time t_j and has destination $x_{i_j+r_j}$.

A queue line operates like an initialised PCS. We denote the delay suffered by X_j by δ_j. Clearly X_j will leave the system at time $t_j + r_j + \delta_j$.

Fact 1 If a queue line S contains m packets then no packet is delayed by more than $m-1$.

Proof Suppose that packet X in S has the worst delay possible in any m-packet line, and w.l.o.g. that all packets have destination x_{n+1}. Then the delay suffered by X depends on the queuing discipline invoked by queue q at time t only if X is present in q at time t. Hence, without loss of generality, we can assume the following queuing discipline: If at time t X is in q_i then for all $t' > t$ and all $j > i$, queue q_j will give preference at time t' to packets that entered at x_k for some $k < j$, over any packets entering at x_j. Clearly, under this assumption, any packet that has moved at least one step in the system after entering, and is then "ahead" of X, will flow at constant speed and will never suffer any later delays.

We choose any packet Y distinct from X. We say that "Y delays X at x_i at time t" if at instant t both X and Y are in q_i and Y is chosen to be transmitted to x_{i+1} in the interval $(t,t+1)$. Suppose now that Y does delay X at x_i at time t. Then at time $t+1$ X will be at x_{i+1}. Since Y will never be delayed again X and Y will never be in the same queue again. Hence Y cannot delay X a second time.

Since there are just $m-1$ choices of Y the result follows. □

The remaining results we need concern the estimation of probabilities. Consider N independent Bernoulli trials each with probability p of success. The probability that at least m of the trials succeed is denoted by $B(m,N,p)$. We shall be interested in the more difficult case of unequal probabilities. The following Theorem of Hoeffding [6] is crucial to our analysis.

Fact 2 If we have N independent Poisson trials with respective probabilities, $p_1,\ldots,p_N$ where $\Sigma p_i = Np$ and if $m \geq Np + 1$ is an integer then the probability of at least m successes is at most $B(m,N,p)$. □

Finally we need bounds on $B(m,N,p)$ itself:

Fact 3 If $m \geq Np$ is an integer then

$$B(m,N,p) \leq \left(\frac{Np}{m}\right)^m \left(\frac{N-Np}{N-m}\right)^{N-m}$$

$$\leq \left(\frac{Np}{m}\right)^m \cdot e^{m-Np}.$$

Proof The first inequality is due to Chernoff [4]. The second follows immediately from the inequality $(1 + x^{-1})^x < e$ in the case that $x = (N-m)/(m-Np)$. □

For m near to Np the following result derived from Chernoff's bound is sometimes useful [1].

Fact 4 If $m = Np(1+\beta)$ where $0 \leq \beta \leq 1$ then $B(m,N,p) \leq e^{-\frac{1}{2}\beta^2 Np}$. □

4. A Scheme for the Boolean n-Cube

We describe a scheme that has the Boolean n-dimensional cube as its graph. It shares with our later schemes the following two-phase strategy. In Phase A it sends each packet to a randomly chosen node of the graph, in Phase B it sends it to its correct destination.

(i) The graph, defined for powers of two, $N=2^n$, is $G=(V,E)$ where $V = \{0,1,\ldots N-1\}$ and $E = \{(x,x//i) \mid x \in V,\ i \in \{1,\ldots,n\}\}$. Here $x//i$ denotes the number whose binary representation is the same as that of x except in the i^{th} most significant bit.

(ii) The routing algorithm: In each phase each packet X has a ticket T_X that is a vector of length n and executes the following program:

```
cobegin for i:=1 step 1 until T_X[i+1]=0 or i=n
        do Transmit X from current node y
           to y//T_X[i];
coend
```

Clearly, T_X specifies the sequence in which the dimensions are to be traversed. The packets execute the above algorithms simultaneously as fast as the queues allow. Their routes are predetermined by their tickets. The rate at which a packet progresses on its route does, of course, depend on how often other packets share segments of its route.

(iii) Initialization of tickets. We need two subroutines: If C is a set "Pick $c \in C$" means "choose an element of C at random, with each member having the same probability of being chosen, and assign it to c". Given a vector T[1..n] "Pack T" means "if there are r non-zero elements in T assign these to T[1],...,T[r] in order of occurrence and assign zero to T[r+1],...,T[n]". For each packet X its ticket for Phase A is precomputed as follows:

```
begin  for i:=1 step 1 until n do
     begin T_X[i]:=i;
           Pick α∈{0,1};
           if α=0 then T_X[i]:=0;
     end;
     Pack T_X;
end.
```

The effect of running Phase A with such tickets is to send each packet to a randomly chosen node. The choices for the packets are independent (and, in general, not distinct). For each packet X that is sent to node u by Phase A and has destination v, a ticket is precomputed for Phase B as follows:

```
begin  for i:=1 step 1 until n do
       if u^i=v^i then T_X[i]=0 else T_X[i]=i;
       Pack T_X;
end
```

It should be clear that running Phase A to completion and then running Phase B constitutes a correct routing algorithm. In the next section we will show that the two phases when run consecutively are fast with enormous probability. Surprisingly, the proof depends only on three general properties already defined. These we now verify.

Both phases are oblivious since the route of each packet depends only on random decisions in Phase A, and only on its current position relative to its destination in Phase B. Each phase is nonrepeating since if two routes share an edge and then diverge, on dimension i say, then the remainder of these routes will always differ in the i^{th} dimension. Finally we have to verify that if the initial conditions IC place exactly h packets at every node and every destination appears on exactly h packets (i.e. a full h-relation) then each phase is symmetric. Consider an arbitrary edge (u,v) where $u^i \neq v^i$. The packets that pass through (u,v) in Phase A are just those that originate for some x such that $x^j=u^j$ for every j with $i \leq j \leq n$. There are $h2^i$ such packets. Each one has probability 2^{-i} of being at u after it has traversed dimensions $1,\ldots,i$ as directed by its ticket. Since each packet has probability one half of then traversing (u,v) we conclude that the expected number of packets that pass along this edge is $h2^i.2^{-i}.2^{-1}=h/2$. Hence the initialised scheme Phase A is symmetric. Phase B is run with input supplied by Phase A. The argument that Phase B is symmetric follows by a similar arguement.

The results of the next section show that the above three properties guarantee termination of each phase within time $K.\log_2 N$ (K independent of N) with overwhelming probability. They also show that the same applies if some of the packets are removed so as to get a partial h-relation. It is easy to see that claims about the phases apply also to the overall algorithm when implemented in the following way: Run Phase A; start running Phase B for each packet either at time $K.\log_2 N$, if the packet has finished in Phase A by then, or as soon as it has finished in Phase A. The algorithm is still correct and with overwhelming probability runs in two time-disjoint phases as intended.

5. The General Theorem

We shall first state the general result and then apply it to the scheme of the previous section.

Theorem 1 In any initialised scheme that is oblivious, nonrepeating and symmetric, and has (i) N nodes, (ii) degree d, (iii) $T = hN$ packets in total, (iv) maximal route length μ, and (v) expected route length η, the probability that some packet is delayed by at least Δ units is less than

$$\left(\frac{e\eta\mu h}{\Delta d}\right)^{\Delta}.T$$

where $e = 2.71\ldots$

Example 1 The scheme based on the n-cube has $d = \mu = n$ and $\eta = n/2$ for each phase, where $N = 2^n$. Consider the case of permutations (i.e. $h = 1$). The probability that a phase fails to finish within time $\mu + \Delta = n + Kn$ is at most

$$\left(\frac{e}{2K}\right)^{Kn}.2^n$$

Hence for all $K \geq 2.5$ this probability is at most

$$4^{-Kn}.2^n \leq N^{-K}.$$

In other words the probability of not finishing in time $(K+1)\log_2 N$ vanishes rapidly as K increases.

Proof of Theorem 1 Packet X intersects edge e_i in a run of the scheme if its route contains e_i. Consider some fixed route R and name the edges of G

$\{e_1,\ldots,e_{Nd}\}$ so that $R = e_1 e_2 .. e_r$. Let P_X be the probability that X intersects at least one edge in R. Then

$$P_X \le \sum_{i=1}^{r} P_{Xi} \qquad (1)$$

where P_{Xi} is the probability that X intersects edge e_i. Since $Traff(e_i) = \sum_X P_{Xi}$ it follows from the symmetry assumption that

$$\sum_X P_{Xi} = \sum_X P_{Xj} \quad \text{for all} \quad i,j \quad (1 \le i,j \le Nd).$$

Therefore from (1)

$$\sum_X P_X \le r \cdot \sum_X P_{X1} \qquad (2)$$

and the total expected message passing is

$$\begin{aligned} h\eta N &= \sum_{i=1}^{Nd} Traff(e_i) \\ &= \sum_{i=1}^{Nd} \sum_X P_{Xi} \\ &= Nd \sum_X P_{X1} \qquad (3) \end{aligned}$$

Relations (2) and (3) imply directly that

$$\sum_X P_X \le rh\eta/d \qquad (4).$$

Now consider a particular packet Y and let R be the route taken by Y in a particular run of the scheme. Then amongst the remaining $T-1$ packets in the system the expected number whose paths intersect R is clearly less than $rh\eta/d$. We therefore have T independent Poisson trials with probability summing to less than $rh\eta/d$. (N.B. For simplicity we are adding a dummy trial having zero probability of success.) Independence among the trials is guaranteed by obliviousness. By Hoeffding's Theorem (Fact 2) the probability of having at least Δ successes is bounded above by

$$B(\Delta, T, r\eta/Nd)$$

provided $\Delta > hr\eta/d + 1$. Fact 3 bounds this above by

$$\begin{aligned} &\left(\frac{rh\eta}{d\Delta}\right)^{\Delta} . e^{\Delta} \\ &= \left(\frac{erh\eta}{d\Delta}\right)^{\Delta} . \qquad (5) \end{aligned}$$

Hence for any fixed packet Y expression (5) bounds the probability that Δ other packets intersect with the route of Y. By Fact 1 the probability of Y suffering a delay of Δ is bounded by this very same quantity.

Since there are T packets to consider the probability that at least one of them suffers a delay of at least Δ is bounded by T times this quantity. □

Corollary 1 Suppose (S,IC) is an initialized scheme satisfying all the hypotheses of Theorem 1. Suppose that a subset of the packets in (S,IC) are simply deleted at the start to give (S,IC'). Then the probability that at least one packet is delayed Δ or more units in a run of (S,IC') is less than

$$\left(\frac{e\eta\mu h}{\Delta d}\right)^{\Delta} . T$$

where $e = 2.71....$

Proof Relation (4) was established for (S,IC). Since the algorithm is oblivious P_X is not affected by the removal of packets other than X. Hence (4) holds for (S,IC') also. The remainder of the Proof of Theorem 1 then applies. (If w packets are removed then for simplicity we now add $w+1$ dummy Poisson trials having zero probability of success.) □

We note that for most reasonable schemes it is "intuitively obvious" that the deletion of any packets must, if anything, make the scheme faster. Unfortunately we know of no simple criterion that provably guarantees this. What the proof of Corollary 1 shows is that provided a relation such as (4) in Theorem 1 holds and the scheme is oblivious and nonrepeating then this does indeed hold. (N.B. Symmetry is not necessary here.)

Example 2 Consider the n-cube scheme of Section 4 when realising a partial h-relation. Then $d = \mu = n$ and $\eta = n/2$. The probability that a phase fails to finish within time $n + Khn$ is at most

$$\left(\frac{e}{2K}\right)^{Khn} . h2^n$$

This is less than $h.N^{-Kh}$ for all $K \ge 2.5$. Hence completion within time $(Kh+1)\log_2 N$ is assured with large probability for K appropriately large.

6. The d-Way Shuffle

Experiments with an earlier algorithm for the n-cube suggested that the run-time distribution was much more sharply peaked around n than could be established by analysis [16]. This was not entirely surprising since the analysis charges for each pair of intersecting routes without allowing for the fact

that these often occur at different times and hence cause no delays. The results did strongly suggest, however, that the real delays were so small that the most important factor in the run-time of a scheme was simply the diameter of the graph. In this respect the cube is non-optimal since with degree $d = \log_2 N$ we may expect every node to be reached from every other within distance $\log_d N = \log_2 N / \log_2 \log_2 N$ rather than $\log_2 N$ as with the cube. With this motivation we considered the following family of schemes, called the d-way shuffle, indexed by the parameter d.

(i) The graph, $G = (V,E)$ is defined for $N = d^n$. The nodes are $V = \{0,\ldots,N-1\}$ and the directed edges

$$E = \{(x,y) \mid x^{k-1} = y^k \text{ for each } k \ (2 \le k \le n)\}$$

where for $z \in V$ z^i denotes the i^{th} most significant digit in the d-ary representation of the number z. In other words there is an edge from x to y if the last $n-1$ d-ary digits of y equal the first $n-1$ of x . (N.B. If $d = 2$ this graph is closely related to but not identical with what is known as the "shuffle/exchange" graph.)

(ii) The routing algorithm consists of two phases A and B. In each phase each packet X has a ticket vector $T_X[i]$ of length n . In each phase the packets execute the following program simultaneously

```
cobegin for i: = 1 step 1 until n do
    Transmit X from current node y to that
    adjacent node z with z^1 = T_X[i].
coend.
```

(iii) Initialization of tickets.

```
Phase A: begin for i: = 1 step 1 until n do
             begin Pick α ∈ {0,...,d - 1}
                   T_X(i) = α;
             end
         end

Phase B: Suppose that packet X is at node u
         and destined for node v .
         begin for i = 1 step 1 until n
               do T_X[i] : = v^i .
         end
```

It should be clear that Phase A sends each packet to a random node, while Phase B then sends it to its correct destination.

Unfortunately our general Theorem 1 cannot be applied here directly because the algorithm in its present form is neither symmetric nor non-repeating. The problem can be illustrated with $d = 2$ and $n = 4$. For any u and v there is a length four path between u and v. There may, however, be shorter paths also e.g. (1011 → 1101) and (1011 → 1101 → 0110 → 1011 → 1101) are two distinct paths between the same pair of nodes. Indeed the node 0000 has a self-loop. Fortunately we can see easily that given u,v and an integer k $(0 \le k \le n)$ there is at most one path of length k between u and v. Using this fact we can guarantee the algorithm to be non-repeating by modifying the ticket preparation as follows in both phases: If a packet X at node u is given a ticket that sends it to v, modify the ticket so that it still sends X to v but does so by the shortest route. (Given u and v such a ticket can be found easily by trying $k = 0,1,\ldots,n-1$). This new scheme, called the modified scheme for the d-way shuffle, is, of course, still not symmetric. Luckily direct analysis is possible.

In the proof of Theorem 1 a bound (4) is obtained on P_X using symmetry. The rest of that proof assumes only obliviousness and non-repetition. Here we shall give a bound on P_X by direct analysis. The remainder of the argument will follow as before.

Theorem 2 When the modified scheme for the d-way shuffle is used to implement an h-relation, in both phases the probability that some packet is delayed by at least Δ units is less than

$$\left(\frac{ehn^2}{(d-1)\Delta}\right)^{\Delta} . hN$$

where $e = 2.71\ldots$.

Proof First we consider complete h-relations. In either phase we consider some fixed route R and rename the edges $E = \{e_1,\ldots,e_{Nd}\}$ so that $R = e_1 e_2 \ldots e_r$. Let P_X be the probability that packet X intersects at least one edge in R. Then

$$P_X \le \sum_{i=1}^{r} P_{Xi} \qquad (1')$$

where P_{Xi} is the probability that X intersects edge e_i .

(i) Phase A: Consider an edge $e_i = (u,v)$ in R. For $1 \le j \le n$ let

$L_j = \{x \mid$ shortest path from x to u has length $j-1\}$.

Suppose packet X starts at $x \in L_j$. If it intersects e_i then it travels $0,1,2,\ldots n-j-1$, or $n-j$ units after leaving v. Since the outdegree of G is d X must be destined for one of at most

$$\sum_{k=0}^{n-j} d^k \leq d^{n-j+1}/(d-1)$$

out of the $N = d^n$ possible nodes. Since the destinations are randomly chosen with equal probabilities we conclude that

$$P_{Xi} \leq 1/d^{j-1}(d-1).$$

Since the indegree of G is d the total number of packets starting at nodes in L_j is at most hd^{j-1}. Hence

$$\sum_X P_{Xi} \leq \sum_{j=1}^{n} \frac{hd^{j-1}}{d^{j-1}(d-1)} = \frac{hn}{d-1}$$

and from (1')

$$\sum_X P_X \leq \sum_{i=1}^{r} \sum_X P_{Xi} \leq hn^2/(d-1) \qquad (4')$$

(ii) Phase B: The argument is the mirror image of the one above for Phase A. Consider an edge $e_i = (u,v)$ in R. Let

$L_j = \{x \mid$ shortest path from v to x is length $j-1\}$

Suppose packet X is destined for $x \in L_j$. If it intersects e_i then it must have travelled $0,1,2,\ldots n-j-1$ or $n-j$ steps before reaching u. Hence it could have come from any one of at most $d^{n-j+1}/(d-1)$ out of the d^n possible nodes. Also the total number of packets with destinations in L_j is at most hd^{j-1}. Hence, as before,

$$\sum_X P_X = \sum_{i=1}^{r} \sum_X P_{Xi} \leq hn^2/(d-1). \qquad (4')$$

The argument deducing Theorem 1 from relation (4) applies here verbatim except that the factor $\mu n/d$ is replaced here by $n^2/(d-1)$.

Finally we note that if we are implementing a partial rather than total h-relation the argument of Corollary 1 applies here equally. □

Example 3 Consider the d-way shuffle with $d = n$ (i.e. $N = n^n$) when realising a partial h-relation with $h = \log_2 n$. The probability of a phase taking time more than $n + \Delta$ where $\Delta = K\log_2 n \simeq K\log_2 N$ is bounded above by

$$\left(\frac{en^2 \log n}{K(n-1)n\log n}\right)^{K n \log_2 n} \cdot n^{n \log_2 n}$$

$$\left(\frac{2e}{K}\right)^{K n \log_2 n} \cdot 2^{2n\log_2 n}$$

for all $n \geq 2$ and all K. Hence for some K_o for all $K \geq K_o$ the above quantity is at most N^{-K}.

Example 4 Consider the d-way shuffle with $d = n$ and $h = 1$. Then for some K_o the probability of the delay exceeding $\Delta = (K\log_2 n)/\log_2\log_2 n$ is at most $N^{-K/2}$ for all $K \geq K_o$. This shows that $O(\log N/\log\log\log N)$ delay can be achieved using degree $O(\log N/\log\log N)$.

Example 5 Consider the d-way shuffle with $d = n/\log_2 n$ and $h = 1$. Then for some K_o the probability of the delay exceeding $\Delta = K\log_2 n$ is again at most N^{-K} for all $K \geq K_o$. Hence time $O(\log N)$ is achieved with degree $O((\log N)/(\log\log N)^2)$.

We conjecture that the runtime is bounded by $O(\log N)$ even when d is a constant. The experimental results reported in the next section give strong evidence for this.

7. Experimental Results

The algorithms were simulated by PASCAL programs on a VAX 11/780 with the aid of a linear congruential random number generator supplied by the system. Some evidence for the suitability of this generator for the problems in hand was reported in [17].

Each experiment consisted of one hundred simulations. Each simulation was for an individual phase of an individual algorithm. Phase B was always supplied with inputs generated by the corresponding Phase A. The order of packets in each queue that was nonempty when Phase B was about to start, was always randomized prior to running Phase B. All the experiments recorded here were for the identity permutation. Other permutations were tried occasionally as spot checks to verify consistency. In each case the "first-in-first-out" queuing discipline was used. The algorithms simulated were the following:

CUBES: The algorithm for the n-cube described in section 4.

BASIC: The algorithm for the n-cube described in [16,17]. It is the same as CUBES except that there

is a final operation in ticket preparation after "Pack": randomize the order of nonzero elements.

CUBESS: As CUBES but before "Pack" in ticket preparation perform the following: Pick $k \in \{0,\ldots,n-1\}$, shift T_x cyclicly to the right by k.

d-SHUFFLE: The (unmodified) scheme described in Section 6.

d-MSHUFFLE: The modified scheme described in Section 6.

CCC: The graph is the cube-connected cycles [9] with each edge directed both ways (i.e. $d = 3$). Suppose cycles are of length s and there are $N = s2^s$ nodes. Ticket preparation is essentially as in CUBES except that a node on the target cycle has to be specified also (i.e. randomly in Phase A, according to destination in Phase B.) In between traversing the dimensions as specified by the ticket the packet traverses edges of cycles in a predetermined (e.g. clockwise) direction. (N.B. A total of s such cycle-edge steps is always sufficient.) On arriving at the target cycle the packet takes the shorter path to its target (i.e. up to s/2 further steps.)

SHUFFLE/EXCHANGE. The graph has $N = 2^n$ nodes $\{0,\ldots,N-1\}$, and the edges consist of (i) shuffle edges (x,y), where $y = 2x$ if $x < N/2$ and $y = 2x + 1 - N$ if $x \geq N/2$, and of (ii) exchange edges (x,y) where if x is even then $y = x + 1$, and if x is odd then $y = x - 1$. Ticket preparation for phase A consists of generating a random $0-1$ vector of length n. The r^{th} digit indicates whether the r^{th} bit in the binary representation of the start address has to change in the phase. Routing a packet may be regarded as an n-stage left shift on the binary representation of this address, with the destination address being shifted in at the least significant binary digit. The packet therefore proceeds through n stages, each consisting of a shuffle step and, optionally, an exchange step.

Figures 1,2 and 3 record the mean values obtained for each of the following three measures: (a) Runtime of Phase A, (b) Runtime of Phase B, and (c) maximal node population in Phase A (i.e. the maximal number of packets that ever reside at one node at one integral time instant in the simulation). The last measure is clearly important in determining the amount of fast memory needed at each node of an implemented PCS. The graphs show the following schemes for each relevant value of N in the range $10 < N < 5000$: CUBES, CCC, SHUFFLE/EXCHANGE, and d-SHUFFLE and d-MSHUFFLE for various values of d.

We summarize some other features of the experimental results as follows:

(i) In no experiment on CUBES, BASIC, CUBESS, 3-SHUFFLE,...,8-SHUFFLE, 3-MSHUFFLE or 4-MSHUFFLE was the variance in either runtime measure (a) or (b) greater than 0.6. For 2-SHUFFLE and 2-MSHUFFLE it was never greater than 1.1.

(ii) In no experiment on either phase of any algorithm was the variance in (c) greater than 0.7.

(iii) CUBES, BASIC and CUBESS had broadly comparable performance. In Phase A CUBES and CUBESS both outperformed BASIC as far as time (!?) but for node population CUBESS was best and CUBES worst of the three. For runtime in Phase B BASIC was better than both CUBES and CUBESS.

(iv) The maximal node population in Phase B was always found to be less than or about equal to that in Phase A.

(v) For d-SHUFFLE a "furthest-to-go-first-out" queuing discipline was tried but this gave only a negligible improvement (N.B. For BASIC a substantial improvement was found [17]).

The main conclusion of these experiments is that the d-way shuffles give outstanding performance even when the degree d is a small constant.

8. Routing in Square Grids

In previous sections we assumed that transmission times for all wires were identical. Here we shall consider the opposite extreme, the case when transmission times are proportional to wire length. Under these circumstances a regular array of processors is a very reasonable interconnection pattern. We first give an algorithm for realising permutations in a two-dimensional grid. Later we generalise this to k-dimensional grids, for arbitrary k. Finally we show that the algorithms need storage for only $O(\log N)$ packets at each node. The scheme for two dimensions is as follows.

(i) The graph is $G=(V,E)$ where $V=\{[i,j] \mid 0 \leq i < n, 0 \leq j < n\}$ and

$E = \{([i,j],[l,m]) \mid (i=l \text{ and } j=m\pm1) \text{ or } (j=m \text{ and } i=l\pm1)\}$

Clearly there are $N=n^2$ nodes. They all have degree

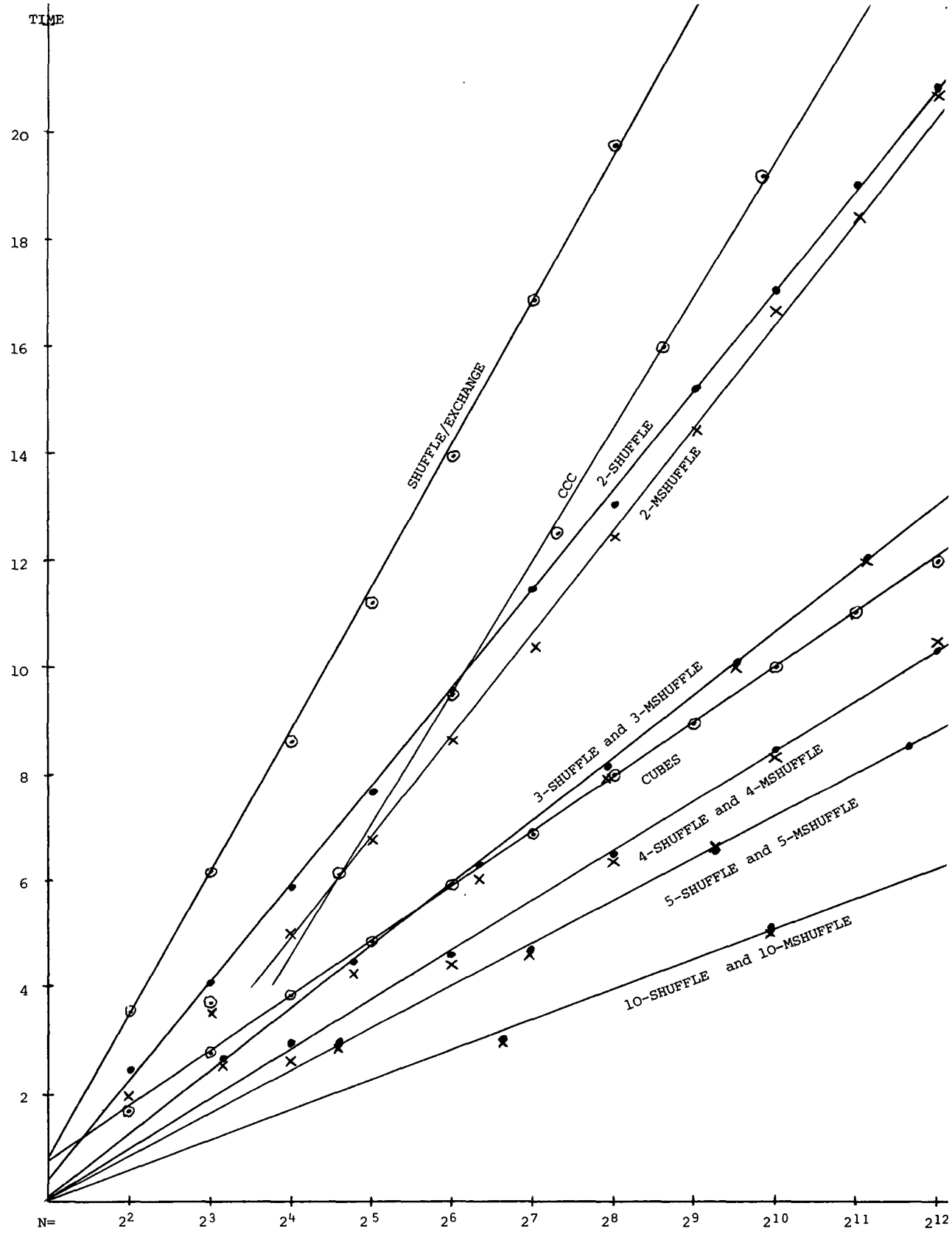

Figure 1. Completion times for Phase A: mean values for one hundred runs.

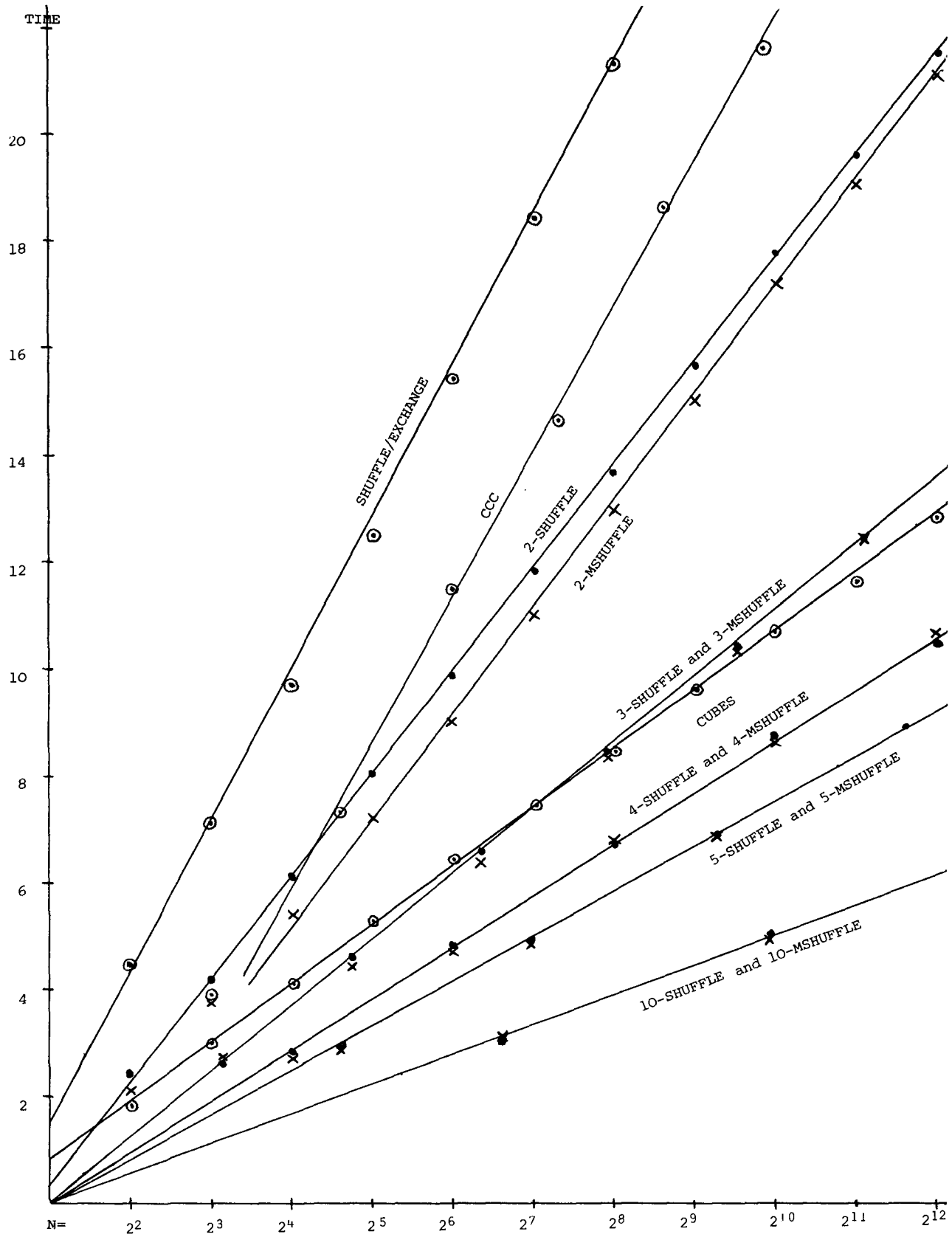

Figure 2. Completion times for Phase B: mean values for one hundred runs.

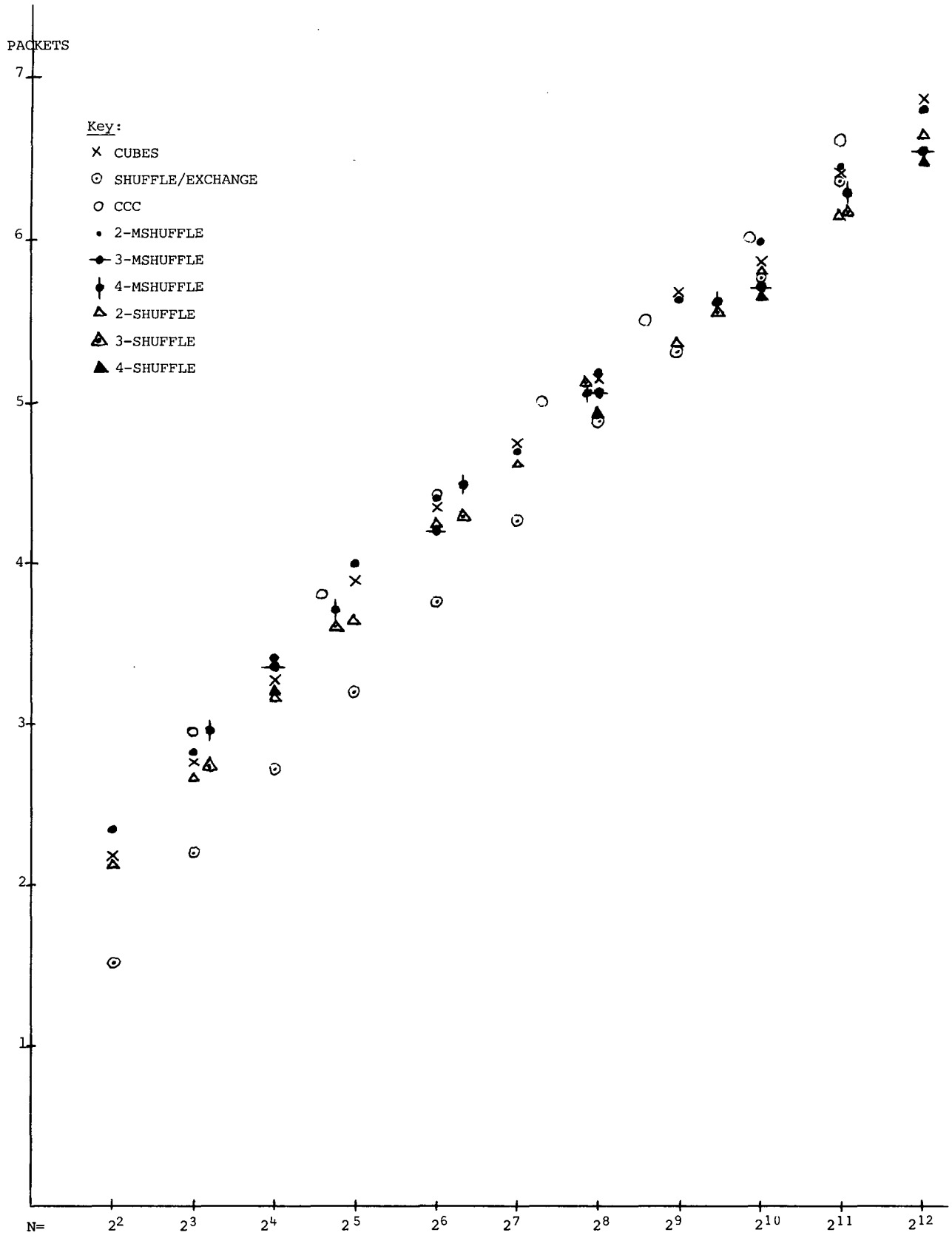

Figure 3. Maximal node populations in Phase A: mean values for one hundred runs.

four except those on the boundaries.

(ii) The algorithm consists of three phases. Phase A randomizes in the first (say vertical) dimension Phase Z then corrects in the second dimension. Finally Phase B corrects in the first dimension.

Phase A For a packet at (i,j) Pick $k \in \{0,\ldots,n-1\}$ and send the packet along the column to (k,j).

Phase Z For a packet at (k,j) destined for (l,m) send it along the row to (k,m).

Phase B For a packet at (k,m) destined for (l,m) send it along the column to (l,m).

Since there is one packet initially at each node Phase A can be implemented by a continuous flow of packets along columns involving no delays whatsoever.

In Phases Z and B any queuing discipline with the following property suffices: "packets that have already moved in the current phase take precedence over any that have not". In these phases, therefore, once a packet has started moving it never suffers a delay since the packets already moving in front of it continue to flow along at a constant speed, while those that have not moved have lower precedence. Hence the delay to a packet starting Phase Z at (k,s) and moving right is at most the total number of packets starting the phase in $\{(k,\bar{j}) \mid \bar{j} \leq s\}$. The corresponding statement holds for packets moving left. Both statements hold for Phase B with respect to columns.

At the start of Phase Z we have the following situation : For each node (k,j) each of the n packets originating in column j has probability $1/n$ of being at $(k,\bar{j})$. Hence for each set of nodes $\{(k,\bar{j}) \mid 1 \leq \bar{j} \leq s\}$ the probability that there are at least $s+g$ packets in total at them at the beginning of Phase Z is $B(s+g, ns, 1/n)$. Since it has at most $n-s$ steps to move, this quantity $B(s+g, ns, 1/n)$ bounds the probability of any one packet taking more than time $(n-s)+(s+g)=n+g$ in Phase Z.

When Phase B starts, each packet is randomly placed in the column of its destination. Hence for each set of nodes $\{(k,m) \mid 1 \leq k \leq s\}$ the probability of having at least $s+g$ packets in total at them at the beginning of Phase B is $B(s+g, n, s/n)$. We conclude, as above, that $B(s+g, n, s/n)$ bounds the probability of any one packet taking more than time $n+g$ in Phase B.

Fact 3 gives the same bounds for both $B(s+g, ns, 1/n)$ and $B(s+g, n, s/n)$, namely

$$\left(\frac{s}{s+g}\right)^{s+g}.e^{g} \qquad (*)$$

If $g > 2s$ then this is less than $(s/3s)^{g}.e^{g} = (e/3)^{g}$. If $2s \geq g \geq s$ then it is less than $(s/2s)^{3g/2}.e^{g} = (e/(2\sqrt{2}))^{g}$.

Fact 4 also gives the same bound for both $B(s+g, ns, 1/n)$ and $B(s+g, n, s/n)$, namely $e^{-g^2/2s}$ for $s \geq g \geq 0$.

Taking $g=Kn^{\frac{3}{4}}$ (say) in each of the above three cases yields a bound on the probability of the time exceeding $n+g = n+Kn^{\frac{3}{4}}$ for all values of s $(1 \leq s \leq n)$. For an appropriate constant $\bar{C} < 1$ in all three ranges this probability is bounded by $\bar{C}^{Kn^{\frac{1}{2}}}$.

Since there are n^2 packets altogether in any one of the three phases the probability that at least one packet takes time more than $n+Kn^{\frac{3}{4}}$ is at most $n^{2}\bar{C}^{Kn^{\frac{1}{2}}}$. We therefore conclude:

Theorem 3 There is a PCS for the $n \times n$ grid graph with the following property: There is a constant $C < 1$ such that when realising any permutation the probability that at least one packet has not finished in time $3n + 2Kn^{\frac{3}{4}}$ is less than $C^{Kn^{\frac{1}{2}}}$. □

The proof applies directly to partial permutations. Also, the generalization to h-relations is obvious.

It remains to show that the above algorithm is merely an instance of a family of schemes that apply to any number of dimensions. Consider the k dimensional cube with each side of length n. Let $G_{n,k}$ be the graph formed by placing a node at each point with integer co-ordinates, and a pair of oppositely directed edges between every pair of such points separated by unit distance. The routing algorithm now works in time $(2k-1)n$ or $(2k-1)N^{1/k}$ where $N=n^k$ is the total number of nodes.

The algorithm is recursive in the number of dimensions. Route (n, k, m) is a procedure for routing (partial) permutations in an m-dimensional subcube of $G_{n,k}$. Denoting the dimensions by $1,2,\ldots,k$, a subcube of $G_{n,i}$ is any set of n^i nodes obtained by fixing a final segment

$i+1,\ldots,k$ of the dimensions $1,2,\ldots,k$

The overall algorithm consists of a single call of Route (n,k,k) with one packet initially at each node, all with distinct destination addresses. The action of the routine on a packet X present in the relevant subcube is described below:

```
procedure Route (n,k,m)
begin if m=1 then send X in dimension 1 so as to
        agree in dimension 1 with its destination,
  else begin (Phase A): Send X along dimension m
        randomly;
        Route (n,k,m-1) for each G_{n,m-1} subcube
        of G_{n,m};
        (Phase B): Send X along dimension m so as
        to agree in dimension m with its destina-
        tion;
      end;
end;
```

The behaviour of Route (n,k,m) can be characterised by the following pair of predicates:

Precondition: For each node x of $G_{n,m}$ each of the n^{k-m} packets originating at nodes of $G_{n,k}$ with dimensions $1,\ldots,m$ agreeing with x has probability n^{m-k} of being present at x. (The rest have probability zero of being at x). For any one choice of $G_{n,m}$ the n^k probabilities so defined are mutually independent.

Postcondition: For each node x of $G_{n,m}$ each of the n^{k-m} packets whose destination address agrees with x in dimensions $1,2,\ldots,m$, has probability n^{m-k} of being present at x. (The rest have probability zero of being at x.) For any one choice of $G_{n,m}$ the n^k probabilities so defined are mutually independent.

It is immediate that if Route (n,k,k) is called to realise a permutation, the precondition holds before execution. For showing correctness of the overall algorithm we have to verify that the postcondition holds after this call is completed. Now this can be proved easily by induction. First note that for a call of Route (n,k,1) the precondition implies the postcondition. Also, before each call of Route (n,k,m) the precondition holds. Finally we have to observe that if the precondition implies the postcondition for Route (n,k,m) then it does so also for Route (n,k,m + 1), for any m $(1 \le m < k)$.

To bound the probability of Phase A of a call of Route (n,k,m) taking time at least n+g we consider a packet X moving in the positive direction along dimension m in this call, and starting at x with coordinate value s in this dimension. The probability that it will suffer delay at least g will be bounded by the probability that there are altogether at least s+g packets at nodes agreeing with x in all but the m^{th} dimension, and with their m^{th} coordinate equal to at most s. By the precondition this is at most

$$B(s+g,\ sn^{k-m},\ n^{m-k}).$$

By the same argument but using the postcondition we deduce that the probability of delay g to a packet X in Phase B of a call of Route (n,k,m) is at most

$$B(s+g,\ n^{k-m+1},\ sn^{m-k-1}),$$

Both the above expressions are bounded by (*), and hence for some $\bar{C} < 0$ the probability of the runtime exceeding $n+Kn^{\frac{3}{4}}$ for any packet X for any Phase in any call is bounded above by $\bar{C}^{Kn^{\frac{1}{2}}}$. But there are only n^k packets, $1+n+n^2+\ldots+n^{k-1}$ procedure calls, and two phases. Hence the probability that at least one packet fails to finish at least one phase in time $n+Kn^{\frac{3}{4}}$ is at most $2n^{2k}\bar{C}^{Kn^{\frac{1}{2}}} \le C^{Kn^{\frac{1}{2}}}$ for an appropriate $C < 1$.

Theorem 4 There is a PCS for the k dimensional grid graph with $N=n^k$ nodes that has the following property: There is a constant $C < 1$ such that when realizing any permutation the probability that at least one packet has not finished in time

$$(2k-1)(n+Kn^{\frac{3}{4}})$$

is less than $C^{Kn^{\frac{1}{2}}}$.

The PCS that satisfies the Theorem is merely the routing algorithm with each Phase given $n+Kn^{\frac{3}{4}}$ time. If a Phase does not finish in this time an arbitrary correct routing algorithm can be invoked for the remainder of the computation. Generalizations of Theorem 4 to partial h-relations are easy.

From the precondition and postcondition it is also clear that immediately before or after any recursive call of Route in a run of Route (n,k,k),

at any node the probability of there being more than f packets present there is at most

$$B(n^{k-m}, f, n^{m-k})$$

for some m $(1 \le m \le k)$. By Fact 3 this is at most

$$f^{-f}.e^{f-1} \le N^{-K}$$

if $f = K\log_2 N$ and K is large enough. Also at any node at any instant during any call of Route $(n,k,1)$ the estimate holds also (to within a factor) since there can be at most one packet "in transit" that neither started the phase there nor finishes there. Hence for any fixed k an allocation of space for $K\log_2 N$ packets at each node is sufficient with overwhelming probability.

If the storage restriction is relaxed then fast algorithms are easier to find. As an example the reader should construct a deterministic distributed routing algorithm for $G_{n,2}$ that takes time $2n$ but may need space for up to n packets at a node.

Acknowledgements

We are grateful to Martin Fürer for suggesting that Fact 1 holds unconditionally.

References

[1] D. Angluin and L.G. Valiant. Fast probabilistic algorithms for Hamiltonian circuits and matchings. J. of Comp. and Syst. Sci. (1979) 155-193.

[2] K. Batcher. Sorting networks and their applications. AFIPS Spring Joint Comp. Conf. 32 (1968) 307-314.

[3] V.E. Benes. Mathematical Theory of Connecting Networks and Telephone Traffic. Academic Press, New York (1965).

[4] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. Ann. of Math. Stat. 23 (1952) 493-507.

[5] Z. Galil and W.J. Paul. A practical general purpose parallel computer, (this volume).

[6] W. Hoeffding. On the distribution of the number of successes in independent trials. Ann. of Math. Stat. 27 (1956) 713-721.

[7] G. Lev, N. Pippenger and L.G. Valiant. A fast parallel algorithm for routing in permutation networks. IEEE Trans. on Computers (1981).

[8] D. Nassimi and S Sahni Parallel algorithms to set-up the Benes permutation networks. Manuscript, University of Minnesota.

[9] F.P. Preparata and J. Vuillemin. The cube-connected cycles. IEEE Symp. on Foundations of Comp. Sci, 20 (1979) 140-147.

[10] M.O. Rabin. Probabilistic algorithms. In "Algorithms and Complexity", J.F. Traub (ed.) Academic Press, New York, 1976.

[11] J.T. Schwartz. Untracomputers. ACM TOPLAS 2(1980) 484-521.

[12] H.J. Siegel. Interconnection networks for SIMD machines, Computer, June 1979, 57-65.

[13] R. Solovay and V. Strassen. A fast Monte-Carlo test for primality. SIAM J. on Computing 6(1977) 84-85.

[14] H. Stone. Parallel processing with the perfect shuffle. IEEE Transactions on Computers, C-20:2, (1971) 153-161.

[15] C.D. Thomson and H.T. Kung. Sorting on a mesh-connected parallel computer. CACM 20:4 (1977) 263-271.

[16] L.G. Valiant. A scheme for fast parallel communication. Report CSR-72-80, Computer Science Department, Edinburgh University, (1980).

[17] L.G. Valiant. Experiments with a parallel communication scheme. In Proc. of 18th Allerton Conference on Communication Control and Computing, University of Illinois, Oct. 8-10, (1980).