Efficient Dispersal of Information for Security, Load Balancing, and Fault Tolerance

MICHAEL O. RABIN

Harvard University, Cambridge, Massachusetts

Abstract. An Information Dispersal Algorithm (IDA) is developed that breaks a file F of length $L = |F|$ into n pieces F_i, $1 \leq i \leq n$, each of length $|F_i| = L/m$, so that every m pieces suffice for reconstructing F. Dispersal and reconstruction are computationally efficient. The sum of the lengths $|F_i|$ is $(n/m) \cdot L$. Since n/m can be chosen to be close to 1, the IDA is space efficient. IDA has numerous applications to secure and reliable storage of information in computer networks and even on single disks, to fault-tolerant and efficient transmission of information in networks, and to communications between processors in parallel computers. For the latter problem provably time-efficient and highly fault-tolerant routing on the n-cube is achieved, using just constant size buffers.

Categories and Subject Descriptors: E.4 [**Coding and Information Theory**]: *nonsecret encoding schemes*

General Terms: Algorithms, Verification

Additional Key Words and Phrases: Fault tolerance, parallel computers, routing of data, storage of data

1. *Introduction*

The storage and transmission of data files in distributed systems gives rise to significant security and reliability problems. We may assume the availability of a dependable encryption system, but even then various dangers remain. Consider a user who keeps files at a certain workstation. By encrypting the files and taking appropriate care of the encryption/decryption keys, the user ensures the *confidentiality* of the information when other people use the workstation, or if someone removes the local disk. But under these circumstances the files may still be erased, physically destroyed, or removed.

Consider a distributed network of computers and workstations. The nodes are often connected by a sparse system of physical links where each edge, by definition, directly connects two nodes, but not every pair of nodes are directly linked. A user who wishes to send a file F from node A to some other node B selectes a path π of edges that together link A to B and sends F, possibly in encrypted form, over that path. We assume that error-correction codes are used so that transmission is error free, but there is a certain small probability for edges to go down, thereby causing loss of message. The probability for any particular edge to fail is small, but since a path π from node A to node B may contain many edges, the probability for a path failing is nonnegligible.

Author's address: Harvard University Center for Research in Computing Technology, Aiken Computation Laboratory, Cambridge, MA 02138.

An obvious countermeasure against loss of files is to store copies at other network nodes. If we consider, say, a total of five copies to be necessary for the required safety margin, then this entails a fivefold blowup of the total storage in the system.

In the case of file transmission, we can send a file along a single path π, request a confirmation, and retransmit along a different path in case of failure. This entails a loss in time. Alternatively, we can create k copies of the file, select k paths $\pi_1, \ldots, \pi_k$ connecting node A to node B, and send a copy along each path. This will result in k-fold increase in network load.

We propose a method, the Information Dispersal Algorithm (IDA), of reliably dispersing the information in F into n pieces or locations. The file can be reconstructed from any m pieces. The salient point is that each piece is of size $|F|/m$, where $|F|$ is the size (number of characters) of F. Consequently, the total number of characters is $(n/m) \cdot |F|$. Since we can choose n so that $n/m \sim 1$, our dispersal method is space efficient. Also, the dispersal and reconstruction are computationally efficient.

Shamir's algorithm [11] for sharing secrets breaks a secret string $\mathscr{S}$ into n pieces $\mathscr{S}_1, \ldots, \mathscr{S}_n$, each of the same size as $\mathscr{S}$, $|\mathscr{S}| = |\mathscr{S}_i|$, $1 \leq t \leq n$, so that $\mathscr{S}$ can be reconstructed from any m pieces. Thus his method leads to n-fold increase in total storage, and for the applications in this paper is no better than n-fold replication. On the other hand, in Shamir's scheme, any $m - 1$ pieces give no information about $\mathscr{S}$.

Our method can be viewed as belonging to the field of error correction codes [2], in which extra bits are added to a message creating a block, so that after occurrence of any k errors within the block, the message can still be reconstructed. These k errors can be anywhere in the block, changing a bit or erasing it. We, however, treat a specialized case of the problem and consequently achieve optimal efficiency in data overhead and utter simplicity in the coding–decoding algorithms. Namely, assume we take, say, a file F of 10,000 characters and disperse it into 14 pieces, any 10 of which suffice for reconstructing F. The loss of four identified pieces is equivalent, from the block encoding point of view, to the loss (not mutation) of four characters in the same identified positions within each block. As a result we can do with pieces of size $|F|/10 = 1000$, which means that any surviving 10,000 characters suffice to reconstruct F, that is, optimal efficiency in data overhead. Furthermore, this allows extra efficient and practical encoding–decoding.

Chaum has drawn my attention to [1], where Asmuth and Blakely give another algorithm for information dispersal. Their algorithm uses number theoretic constructs (Chinese remaindering) and is more complicated than the one presented here. They treat specific small values (n, m); it is not clear that optimal space efficiency is obtainable for arbitrary (n, m), as in the present paper.

Coming back to the security–reliability problems, we consider the storage problem, and assume that no more than, say, four nodes will crash, or be taken over by adversaries, between computer sessions of a user. At the end of a session, the user will encode any newly created, or altered, file F. The user will split it into 14 pieces $F_1, \ldots, F_{14}$ so that any 10 pieces suffice to reconstruct F, and store each F_i at a different node. On coming back, the user can find 10 intact pieces from which to reconstruct F. Since $|F_i| = |F|/10$, the total number of characters stored is $|F|{:}14/10$. Thus, the storage-space overhead is just 40%. The numbers $n = 14$, $m = 10$, used here are just for illustration. Our file dispersal scheme is general and the parameters n, m can be chosen as needed.

Similarly for the transmission problem. If we wish to transmit F from node A to node B, we split F as above, select, say, 14 paths $\pi_1, \ldots, \pi_{14}$ from A to B, and send F_i from A to B along π_i, $1 \leq i \leq 14$. Again, if no more than four paths break down, then F will be reconstructable at B, and the blow-up in the total number of transmitted characters will be 40%. An additional benefit from this dispersal is a more even distribution of network load.

In Section 4 we give a detailed application of the Information Dispersal Algorithm (IDA) to the routing of data in parallel computers. We consider a parallel architecture PC_n, where $N = 2^n$ processors are placed at the vertices of the n-dimensional Boolean cube $C_n = \{0, 1\}^n$. Every node $x \in C_n$ is connected by two-way links to its n neighbors. At every node there is a packet Px of information, and we want to simultaneously route every packet Px to a destination $\pi(x) \in C_n$, where $\pi \in S_N$ is a permutation of C_n. In [13], Valiant proposes a randomized algorithm with the following properties. For $K \geq 2.5$, the probability that the total transmission round, that is, arrival of all packets at their destinations, will not terminate in $2(K + 1)\log_2 N$ time units is shown to be smaller than $N^{-(2K-1)}$. The local queues, in which packets await their transfer to neighboring nodes, will not contain more than $K \cdot \log_2 N$ packets, with the same probability. In [6], Pippenger showed that fixed-size queues suffice, at the expense of lengthening the transmission time.

We employ the IDA to achieve assured fast transmission time, small fixed (i.e., independent of N) buffer size, and high fault tolerance. Thus, in Theorem 1, we show that with local buffers containing up to $p = 6$ packets, and total transmission time never exceeding $2 \cdot \log_2 N$, the probability of all packets arriving is at least $1 - N^{-4}$. The exponent of N in this analysis decreases rapidly with p, while the transmission time remains $2 \cdot \log_2 N$. Thus for $p = 9.6$, the probability of all packets arriving is at least $1 - N^{-13}$. As to fault tolerance, if up to N/n links are allowed to fail, and we use the IDA with $m = \lfloor n/2 \rfloor$, the probability that no packets will be lost in a transmission round is at least $1 - N^{-\alpha}$, where $\alpha \sim 0.25 \cdot \log_2 n$ (Theorem 2).

The efficacy of the IDA approach in assuring reliability is brought out by a detailed analysis for the case $n = 10$, i.e., 1024 nodes. Assume that in a time interval T (say T equals one day), ten links are expected to fail and that inspection and repair at the end of T ensure that no more than ten failures accumulate.

If we employ simple duplication of packets, then the probability of loss of packets in a one transmission round due to ten random link faults is 0.32.

If we employ the IDA with $m = 5$, $n = 10$, thus again just doubling the volume of information, then the corresponding probability of failures is less than $7.7 \cdot 10^{-6}$. In terms of error-free intervals, in the simple duplication mode, one interval T in 3 will have some transmission failures. With the IDA, just one interval T in 130 will have some transmission failures.

These quantitative results concerning packet transmission in the n-cube, demonstrate the efficacy of the IDA in load balancing (as evidenced by the small constant buffer size), speed of transmission, and fault tolerance. Similar benefits will occur in applying the IDA to other network topologies.

Other applications of the IDA, to be discussed elsewhere, include simple, efficient, and fault-tolerant broadcasts in ether and satellite-type networks, and dispersal of files in magnetic disk systems as protection against failures.

2. *Splitting and Recombining Files*

Let $F = b_1, b_2, \ldots, b_N$ be a file, that is, a string of characters. Assume that we want to disperse F, either for storage or for transmission, under the given condition that with overwhelming probability no more than k pieces will be lost through node or communication-path failures.

The characters b_i may be considered as integers taken from a certain range $[0 \cdots B]$. For example, if the b_i are eight-bit bytes, then $0 \le b_i \le 255$. Take a prime $B < p$. For bytes, $p = 257$ will suffice; but we may wish to choose a prime larger than the smallest $B < p$. Note that with $p = 257$ there is an excess of one bit per byte, we shall see later how to implement IDA in fields $GF(2^{ss})$, $s = 8$ for bytes, without any excess. Now, F is a string of residues mod p, that is, a string of elements in the finite field Z_p. All the following computations are in Z_p, that is, mod p.

Choose an appropriate integer m so that $n = m + k$ satisfies $n/m \le 1 + \epsilon$ for a specified $\epsilon > 0$. Choose n vectors $a_i = (a_{i1}, \dots, a_{im}) \in Z_p^m$, $1 \le i \le n$, such that every subset of m different vectors are linearly independent. Alternatively, it will suffice to assume that with high probability, a randomly chosen subset of m vectors in $\{a_1, \dots, a_n\}$ is linearly independent. We shall see later on how to satisfy each of these conditions.

The file F is segmented into sequences of length m. Thus

$$F = (b_1, \dots, b_m), (b_{m+1}, \dots, b_{2m}), \dots .$$

Denote $S_1 = (b_1, \dots, b_m)$, etc. For $i = 1, \dots, n$,

$$F_i = c_{i1}, c_{i2}, \dots, c_{iN/m},$$

where

$$c_{ik} = a_i \cdot S_k = a_{i1} \cdot b_{(k-1)m+1} + \cdots + a_{im} \cdot b_{km}. \tag{1}$$

It follows that $|F_i| = |F|/m$.

If m pieces of F, say, $F_1, \dots, F_m$ are given, we reconstruct F as follows. Let $A = (a_{ij})_{1 \le i,j \le m}$ be the $m \times m$ matrix whose ith row is a_i. It is readily seen that

$$A \cdot \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix} = \begin{bmatrix} c_{11} \\ \vdots \\ c_{m1} \end{bmatrix},$$

and hence

$$\begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix} = A^{-1} \cdot \begin{bmatrix} c_{11} \\ \vdots \\ c_{m1} \end{bmatrix}.$$

Denote the ith row of A^{-1} by $(\alpha_{i1}, \dots, a_{im})$, then in general, for $1 \le k \le N/m$,

$$b_j = \alpha_{i1} c_{1k} + \cdots + \alpha_{im} c_{mk}, \qquad 1 \le j \le N, \tag{2}$$

where $i = j \bmod m$, $k = \lceil j/m \rceil$ (here we take the residues to be $1, \dots, m$).

Thus we invert A once and for all, and reconstruct F by (2), which involves $2m$ mod p-operations per character of F. For sufficiently large files satisfying $m^2 \le |F|$, the operation cost of computing A^{-1} is majorized by the cost of reconstructing F by (2), even if we use m^3 operations for computing A^{-1}. It will be shown that we can choose $a_1, \dots, a_n$ so that the computation of any A^{-1} will require just $O(m^2)$ operations.

Since both splitting up the file by (1), and reconstruction by (2) involve just inner products, the method is readily adaptable to vectorized, systolic, or parallel architectures.

Remark. It is possible to use other fields instead of Z_p. Thus, for example, for 8-bit bytes we can directly use the field $E = GF(2^8)$ of characteristic 2 and having 256 elements. All we need is an irreducible polynomial $p(x) \in Z_2[x]$ of degree 8 to allow us to effectively compute in E (see [7]).

2. *The Independence Condition*

Let $x_1, \ldots, x_n, y_1, \ldots, y_m \in Z_p$ satisfy the conditions: For all i and j

$$x_i + y_j \neq 0; \qquad i \neq j \rightarrow x_i \neq x_j \quad \text{and} \quad y_i \neq y_j, \tag{3}$$

(this requires $n + m < p$). Define

$$a_i = \left(\frac{1}{x_i + y_1}, \ldots, \frac{1}{x_i + y_m}\right), \qquad 1 \leq i \leq n.$$

Let A be the matrix with rows $a_1, \ldots, a_m$ then [5, p. 35]

$$|A| = \frac{\prod_{i<j}(x_i - x_j)(y_i - y_j)}{\prod_{i,j}(x_i + y_j)}. \tag{4}$$

It follows from (3) and (4) that $|A| \neq 0$. Thus any m vectors in $\{a_1, \ldots, a_n\}$ are linearly independent.

Furthermore, if $A^{-1} = [b_{ij}]$, then $b_{ij} = (-1)^{i+j}|A(j, i)|/|A|$, where $A(j, i)$ is the matrix obtained from A by deleting the jth row and ith column. Denote, for $1 \leq k \leq m$,

$$c_k = \prod_{\substack{i<k\\k<j}} (x_i - x_k)(x_k - x_j),$$

$$d_k = \prod_{\substack{i<k\\k<j}} (y_i - y_k)(y_k - x_j),$$

$$e_k = \prod_{j} (x_k + y_j),$$

$$f_k = \prod_{i} (x_i + y_k).$$

These quantities can be calculated in $O(m^2)$ operations. Now (4) implies

$$|A(j, i)| = \frac{|A|}{c_j \cdot d_i \cdot e_j \cdot f_i} \cdot (x_j + y_j).$$

Thus A^{-1} can be computed by $O(m^2)$ operations.

Another way to ensure that any m vectors out of $a_1, \ldots, a_n$ are linearly independent, is to choose n different elements $a_1, \ldots, a_n \in Z_p$ (this requires $n < p$) and set

$$a_i = (1, a_i, \ldots, a_i^{m-1}), \qquad 1 \leq i \leq n.$$

If we want every a_i to depend on more than one parameter α_i, we can simply choose $a_i = (a_{i1}, \ldots, a_{im})$ *randomly*, by randomly and independently selecting the residues $a_{ij} \in Z_p$. Every $m \times$ m matrix A, obtained by selecting m different vectors out of $\{a_1, \ldots, a_n\}$ as rows, is again a randomly chosen matrix. It is readily seen that

$$1 - \frac{1}{p} \cdot \frac{p}{p-1} \leq \Pr(A: |A| \neq 0) \leq 1 - \frac{1}{p},$$

so that for $100 < p$ with probability nearly $1 - (1/p)$, the matrix is nonsingular.

If we want to increase the probability that the matrix A used in the reconstruction is nonsingular, we can choose larger primes p. Thus, if F is a string of bytes, we can choose a prime p satisfying $2^{16} < p$. Considering every pair $b_1b_2, b_3b_4, \ldots$ of bytes as a character, we employ our scheme. Now the probability of having a nonsingular matrix is at least $1 - 1/64{,}000$.

3. *Fingerprints*

In implementing the file dispersal scheme, it is useful to include the coding vector $a_i = (a_{i1}, \ldots, a_{im})$ as a header of the piece F_i, so that $F_i = d_{i1}\, d_{i2} \cdots d_{iM}$, where $M = N/m + m$, $d_{i1} = a_{i1}$, $d_{i2} = a_{i2}$, $d_{i,m+1} = c_{i1}$, etc. In this way there is no need to store the vectors $a_1, \ldots, a_n$ separately, they also are protected by dispersal.

Unlike Shamir's sharing of secrets, in our scheme $m - 1$ pieces $F_1, \ldots, F_{m-1}$ may provide *some* information about F. For the version employing randomly chosen vectors $a_1, \ldots, a_n$, we do not see a way of fully reconstructing even small portions of F from $m - 1$ pieces. Still, it is possible that an adversary will obtain m pieces of F by eavesdropping. For this reason it is best to encrypt F before dispersing it, and decode it after reconstruction of the encrypted version. This does not change anything in our scheme.

One possible attack on the security of the dispersed storage scheme would be to replace a piece F_i at some node by a string G. If we use G as one of the pieces for reconstruction, we shall not get F back.

A reliable method to counter this replacement attack is to use *fingerprints* [8]. We randomly choose an irreducible polynomial $f(x) \in Z_p[x]$ of a small degree k, say, $k = 11$. We consider the piece (including the header a_i) $F_i = d_{i1} \cdots d_{iM}$, $M = N/m + m$, as a polynomial

$$F_i(x) = d_{i1}x^{M-1} + \cdots d_{iM}.$$

and compute $P_i = res(F_i(x), f(x))$. We encrypt P_i by an encryption function E and store or transmit the pairs $(E(P_i), F_i)$, $1 \le i \le n$. If we use IDA for transmission we must also transmit $E(f)$.

When receiving pairs, call them (H, G), for reconstructing F, we test whether

$$E^{-1}(H) = res(G(x), f). \tag{5}$$

Only if (5) holds for a pair (H, G), do we use G as one of the pieces F_i. By assumption, we shall have m such pieces.

It is proved in [8] that an adversary, who does not know the polynomial $f(x)$, has an exponentially small probability of producing a pair (H, G) for which (5) holds. For $p = 257$, $deg\, f = 11$, and files with up to a million characters, that probability is smaller than 10^{-20}.

4. *Routing for Parallel Computers*

As a typical application of our information dispersal algorithm we consider the problem of routing of information in parallel computers.

Let PC_n be a parallel computer consisting of $N = 2^n$ nodes, where each node x contains a processor Cx and local memory module Mx. We name the nodes by elements of the Boolean n-cube $C_n = \{0, 1\}^n$. For $x \in C_n$, $1 \le i \le n$, the notation $x \,//\, i$ denotes the node $y \in C_n$ such that $x[j] = y[j]$ for $j \ne i$, and $y[i] = 1 - x[i]$. In PC_n the node x is connected by two-way links to each of the nodes $x \,//\, i$, $1 \le i \le n$.

Cube-based architectures for parallel computers were considered in theoretical studies [12], and were actually constructed [4, 10]. A basic question is how information will be routed between the memory modules. In his ground-breaking paper [13], Valiant considers a model where at each node x there is a packet Px of information that has to be sent to a destination node $\pi(x)$, where $\pi\colon C_n \rightarrow C_n$ is a permutation. He proposes a randomized algorithm consisting of two phases. In Phase 1 every node x chooses a random intermediate node $R(x)$ and routes Px from x to $R(x)$. In Phase 2, every packet Px is routed from $R(x)$ to $\pi(x)$. Valiant proves that with overwhelming probability $1 - N^{-k}$, every packet reaches its destination within time $c \cdot \log_2 N$, and queues at the individual nodes contain no more than $d \cdot \log_2 N$ packets. The exponent k depends on c and d.

We consider the case that the packets Px are sufficiently large, say the size of *pages* in a virtual memory operating system, so that the method of information dispersal is applicable. We shall give a routing algorithm employing information dispersal, which will require only fixed-sized queues (buffers) and which will exhibit fault-tolerance to a large number of link failures.

In the spirit of this paper, the packet Px emanating from a node n will be broken into n pieces $Px1, \ldots, Pxn$, so that every $m = \lfloor 5n/6 \rfloor$ pieces suffice for reconstructing Px.

In the transfer of Px from x to $\pi(x)$, each piece Pxi, $1 \le i \le n$, will be routed independently. The piece Pxi will be supplied at x with a *ticket* Txi that is a vector of length $2 \cdot (n + 1)$ of integers $0 \le k \le n$.

At any time $1 \le t \le 2(n + 1)$ during the execution of the simultaneous routing of all pieces Pxi, $x \in C_n$, $1 \le i \le n$, to their destinations $\pi(x)$, every node y contains a number of pieces (P', T'), (P'', T''), ..., where T' is the ticket attached to P, T'' is attached to P'', etc. The processor Cy tests $T'[t]$, $T''[t]$, ..., and for every $1 \le j \le n$ sends *all* the packets $P^{(r)}$ with $T^{(r)}[t] = j$ to the node $y \,//\, j$. This is completed by time $t + 1$ for all nodes $y \in C_n$, and for all links from y to $y \,//\, j$, $1 \le j \le n$. If $T^{(r)}[j] = 0$, then $P^{(r)}$ stays at y at time t.

We also make an assumption concerning the temporary storage available at each node for the packets in transit. Let the size, that is, number of characters, of every packet be $|Px| = L$. Then $|Pxi| = L/m$, $m = \lfloor 5n/6 \rfloor$, and $\sum_i |Pxi| = 1.2 \cdot L$. We assume that at every node $y \in C_n$ there is a buffer $BF(y)$ of size $6 \cdot L$, large enough to accommodate six packets Px.

For the above communications model, we implement the concurrent routing of all packets Px from x to $\pi(x)$, $x \in C_n$, by the following Routing Algorithm:

```
cobegin for $x \in C_n$:
(1) Split $Px$ into $n = \log_2 N$ pieces $Px1, \ldots, Pxn$;
(2) Choose randomly $n$ pairwise different nodes $R_1(x), \ldots, R_n(x)$;
(3) Select pairwise vertex-disjoint (except for $x$) paths $D_1(x), \ldots, D_n(x)$ from $x$ to
    $R_1(x), \ldots, R_n(x)$, each of length at most $n + 1$;
(4) Select vertex disjoint paths $E_i(x)$, $1 \le i \le n$, from $R_i(x)$ to $\pi(x)$, each of length at most
    $n + 1$;
(5) Attach to $Pxi$, $1 \le i \le n$, a ticket $Txi$ for routing from $x$ to $\pi(x)$, along $D_i(x)$ followed
    with $E_i(x)$;
(6) Simultaneously send all pieces $(Pxi, Txi)$ to $\pi(x)$ in the manner explained before.
coend.
```

We shall call an execution of the Routing Algorithm, in which for a permutation $\pi \in S_N$ packets Px are simultaneously routed from x to $\pi(x)$ for every $x \in C_n$, a *transmission round*.

Comment 1. Every $m = \lfloor 5n/6 \rfloor$ of the pieces $Px1, \ldots, Pxn$ suffice to reconstruct Px. Later on, for Theorem 2, we shall take $m = \lfloor n/2 \rfloor$.

Comment 2. A proof that such paths $D_i(x)$, $1 \le i \le n$, exist, and an algorithm for their construction are given in Lemma 3.

Comment 3. For the sake of uniformity we pad a ticket Txi, when $length(D_i(x)) = k < n + 1$, by zeros so that $Txi[k+1] = \cdots = Txi[n+1] = 0$, and similarly for the $E_i(x)$ portion of the route.

Comment 4. In phase 6, at any time $1 \le t \le 2(n+1)$, if a buffer $BF(y)$ at a node y receives more than $5n$ pieces Pxi (with different pairs (x, i)), then the overflow above its capacity of $6 \cdot L$ (the equivalent of six original packets), will be rejected and *lost.*

THEOREM 1. *Under the Routing Algorithm* (*RA*), *for any given permutation* π, *the probability of all packets reaching their destination is at least* $1 - (1/N^4)$.

PROOF. Consider a time point $1 \le t \le 2(n+1)$, and a node y. Denote by $Y(y, x, t)$ the random variable that is the number of pieces Pxi arriving at y at time t. By Comment 3, all the pieces Pxi are simultaneously either on their respective ways along $D_i(x)$ to $R_i(x)$, or on their way along the paths $E_i(x)$ from $R_i(x)$ to $\pi(x)$. The nonintersection property of the paths $D_i(x)$, $1 \le i \le n$, and the same property for the $E_i(x)$, imply that for every $x \in C_n$, at most one piece Pxi arrives at node y at time t. Thus $Y(y, x, t)$ is 0, 1 valued. Denote by $p(y, x, t)$ the probability that $Y(y, x, t) = 1$.

The buffer $BF(y)$ will overflow at time t if

$$\sum_{x \in C_n} Y(y, x, t) \ge 5n. \tag{6}$$

We want to show that the probability for (6) to occur is small. According to Comment 4, the occurrence of buffer-overflow results in the disappearance of pieces Pxi. For the sake of the present proof, we shall assume that no pieces Pxi are lost, that is, we disregard the restriction in Comment 4. The probability $p(y, x, t)$ used in the proof refers to this nondestructive model. Clearly the probability for (6) to occur in this model is greater than the corresponding probability in the actual RA.

The RA itself, and its modified version disregarding overflows, are completely symmetrical at any time $1 \le t \le 2(n+1)$, for all $y \in C_n$. Since at $t = 1$ we have n pieces Pxi at each node, we must have

$$\sum_{x \in C_n} p(y, x, t) = n. \tag{7}$$

From the definition of the RA it follows that the random variables $Y(y, x, t)$ are pairwise independent (here y and t are fixed, $x \in C_n$).

A recent theorem of Raghavan and Spencer [9] states that if $Y_1, \ldots, Y_N$ are independent Bernoulli trials such that the expected value E for their sum is

$$E\left(\sum_i Y_i\right) = n, \tag{8}$$

and if $\delta > 0$, then (with $e = 2.71\ldots$),

$$\Pr\left(\sum_i Y_i \ge (1+\delta)n\right) \le \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^n. \tag{9}$$

The above analysis shows that the conditions of the Raghavan–Spencer theorem hold for the random variables $Y(y, x, t)$, $x \in C_n$. In particular, (7) is condition (8) for our variables. The probability of the buffer-overflow-event (6) is bounded by (9), with $\delta = 4$.

Thus for buffer size $|BF(y)| = 6|Px|$, the probability of overflow of $BF(y)$ at time t is less than $(e^4/5^5)^n \sim 2^{-5n} \cdot 1.8^{-n}$. Now, the probability of overflow of some buffer at some time $1 \le t \le 2(n+1)$, that is, during a transmission round is at most

$$\text{Pr(some overflow during one transmission round)} \le 2^{-5n} \cdot 1.8^{-n} \cdot N \cdot 2(n+1) = N^{-4} \cdot \frac{2(n+1)}{1.8^n}. \tag{10}$$

This implies the theorem for $n \ge 4$.

By way of example, for $n = 10$, the probability, for a transmission round, of even one packet Px not arriving is, according to (10), at most $6 \cdot 10^{-14}$.

We now turn to the study of the fault tolerance properties of the Routing Algorithm on PC_n. To simplify computations we assume that the information dispersal algorithm is used with $m = \lfloor n/2 \rfloor$, so that the total amount of information is doubled. We further assume that the sizes of the buffers $BF(y)$ are a sufficiently large multiple $p \cdot |Px|$ of the basic packet size, so that the probability of loss of a packet due to buffer overflow is negligible.

THEOREM 2. *Assume for PC_n, that within a transmission round fewer than N/n of the links randomly fail. If we break each packet Px into n pieces so that any $m = \lfloor n/2 \rfloor$ pieces suffice for reconstruction of Px, and if we employ the Routing Algorithm with buffer size $|BF(y)| = p \cdot |Px|$ large enough to make the probability of buffer-overflow negligible, then the probability of all packets Px reaching their respective destinations is at least* $1 - 2 \cdot N \cdot (4 \cdot e/n)^{0.25n}$.

PROOF. To realize the permutation $\pi \in S_N$, the RA constructs, for every $x \in C_n$, paths $D_i(x)$ from x to $R_i(x)$ and $E_i(x)$ from $R_i(x)$ to $\pi(x)$, $1 \le i \le n$. The piece Pxi is sent along $(D_i(x), E_i(x))$ from x to $\pi(x)$. The piece Pxi will be lost if one of the faulty links occurs on $D_i(x)$ or on $E_i(x)$.

A packet Px will not be reconstructable at $\pi(x)$, if more than $k = n - m$ of its pieces Pxi are lost. If this happens, then $(k+1)/2$ of the paths $D_i(x)$, $1 \le i \le n$, or $(k+1)/2$ of the paths $E_i(x)$, must have been disconnected by link failures.

We approximate our link failure pattern by assuming that every link has failure probability $(N/n)/nN = 1/n^2$, and that link failures are independent. Consider the event that $(k+1)/2 \ge 0.25 \cdot n$ of the paths $D_i(x)$, $1 \le i \le n$, are disconnected. The probability that a single path $D_i(x)$, which contains at most $n+1$ links (Comment 2), will fail is at most $(n+1)/n^2 \sim 1/n$. Thus the expected number of disconnected paths is $n \cdot 1/n = 1$. By the Raghavan–Spencer relation (9), with $\delta = 0.25n - 1 \sim 0.25n$, and exponent 1,

$$\text{Pr(more than } 0.25n \text{ paths } D_i(x) \text{ fail)} \le \left(\frac{e}{0.25n}\right)^{0.25n} = \left(\frac{4e}{n}\right)^{0.25n}. \tag{11}$$

The same bound holds for the paths $E_i(x)$. By our analysis, the probability that Px will not be reconstructable at $\pi(x)$ is at most twice the probability in (11). Since there are $N = 2^n$ packets, the probability that some packet will fail within a transmission round is at most $2 \cdot N \cdot (4e/n)^{0.25n}$.

Although asymptotically the above probability of failure is smaller than N^{-K} for any fixed K, the generality of our derivation resulted in a formula that does not provide a good estimate for small values of n. It is instructive to carry out a direct, less rigorous, calculation for $n = 10$.

We assume that up to 10 links may break down during a time interval T (say, T equals one day). An inspection and replacement routine ensures that no more than ten links are faulty at any given time. Further, we assume that the paths $D_i(x)$ and $E_i(x)$ are so constructed that also, for the same $x \in C_n$, no $D_i(x)$ has any link in common with any $E_j(x)$. We do not know how to achieve this, but probabilistic considerations suggest that the link-overlap between the union of paths $D_i(x)$ and the union of the $E_j(x)$ can be made small.

Consider first the approach to fault tolerance through simple duplication. Every packet Px, $x \in C_n$, will have two copies $P'x$, $P''x$. The node x will be connected to $\pi(x)$ by two links-disjoint paths G_{x1}, G_{x2}, going as in the Routine Algorithm, through two randomly chosen nodes $R_1(x)$, $R_2(x)$.

The probability of failure of any given link during the time-interval T is $\frac{10}{10{,}000} = \frac{1}{1000}$. Each of the paths $G_i(x)$, $i = 1, 2$, contains about 20 links, so that the probability of a path $G_i(x)$ being disconnected is very nearly $\frac{20}{1000} = \frac{1}{50}$. The packet Px is lost if both copies $P'x$, $P''x$ are lost, that is, if both paths $G_1(x)$, $G_2(x)$ are disconnected; the probability for that is nearly $\frac{1}{2500}$. The probability for all packets to reach their destination is $(1 - (\frac{1}{2500})^{1000} \sim 0.68$. Thus, the probability of some packet failing to arrive at a simple transmission round, is at least 0.32.

Consider now the situation under the Routing Algorithm with $m = 5$, $n = 10$, that is, any five pieces being sufficient for reconstructing a packet. Consider a packet Px and the associated paths $H_i(x) = (D_i(x), E_i(x))$, $1 \le i \le 10$, from x to $\pi(x)$. Again, the probability for a path to be disconnected by a link failure is nearly $\frac{20}{1000} = \frac{1}{50}$. Under our assumption concerning disjointness, the failures of paths, for the same $x \in C_n$, are independent. The packet Px will not be reconstructable at $\pi(x)$ if six or more of the paths $H_i(x)$ fail. Hence

$$\Pr(Px \text{ not reconstructable}) = \binom{10}{6} \cdot \left(\frac{1}{50}\right)^6 \cdot \left(1 - \frac{1}{50}\right)^4 + \cdots.$$

The further terms are negligible, and the first term is smaller than $7.7 \cdot 10^{-9}$. There are $1024 \sim 1000$ packets, so the probability that some packet will fail in one transmission round is at most $7.7 \cdot 10^{-6}$.

This should be compared with the previously computed failure probability of 0.32, when using simple duplication of packets.

The fault-tolerance advantage is further emphasized by considering the behavior throughout a complete time-interval T (for which we assumed no more than 10 random link failures). We modify the Routing Algorithm as follows. Instead of choosing, for the node $x \in C_n$, the intermediate nodes $R_1(x), \ldots, R_n(x)$ randomly, once the permutation $\pi \in S_N$ is given, we randomly choose for every $y \in C_n$, intermediate nodes $R_1(x, y), \ldots, R_n(x, y)$. Furthermore, we select paths $H_i(x, y) = (D_i(x, y), E_i(x, y))$, $1 \le i \le n$, from x to $R_i(x, y)$ to y, in accordance with Lemma 3.

The Modified Routing Algorithm (MRA) will use for realizing a permutation $\pi \in S_N$, for every node $x \in C_n$ the chosen nodes $R_i(x, \pi(x))$, $1 \le i \le n$, and paths $H_i(x, y)$. In practice, the intermediate nodes can be chosen by use of a pseudo-random number generator. Theorem 1 does not hold for the MRA, but it can be shown that for every K, for suitably larger buffer size (still independent of n), with high probability, there will not occur any buffer overflow in N^k transmission rounds.

Coming back to the fault-tolerance question for PC_{10}, we carry out the probability computation for the case of the MRA. For a random failure of 10 links and one choice of intermediate points $R_i(x)$, $1 \le i \le 10$, the probability of failure of the packet Px to arrive is at most $7.7 \cdot 10^{-9}$. Thus the probability of Px not arriving at some $y \in C_{10}$ due to link faults is at most $1024 \cdot 7.7 \cdot 10^{-9} \sim 7.7 \cdot 10^{-6}$. The probability of this happening for some packet Pz, $z \in C_{10}$, is $1024 \cdot 7.7 \cdot 10^{-6} = 7.7 \cdot 10^{-3}$. Since in the MRA we realize every permutation $\pi \in S_N$ by the prechosen path systems $H_i(x, y)$, $1 \le i \le n$, $x, y \in C_n$, the probability that for ten randomly chosen link faults some permutation will not be realized is at most $7.7 \cdot 10^{-3}$.

Now, by our assumption, ten is the expected number of link faults within time T, and inspection and repair (for replacement) keep the total number from exceeding 10. Hence the probability of some packet not arriving in some transmission round in the time-interval T is at most $7.7 \cdot 10^{-3}$. Put differently, in approximately 130 time-intervals T, there will be an expected number of just one time interval with *any* transmission failures due to link faults (and this under the assumption that we allow ten link faults to accumulate). By way of contrast, simple duplication results in expected transmission failures in one out of three time-intervals T.

We must still prove the existence of not-too-long and nonintersecting paths from any $x \in C_n$ to pairwise different $R_i(x)$, $1 \le i \le n$, as claimed in Comment 2. The author had the result with paths of length up to $2n$. Michael Ben-Or (private communication) suggested the improvement to paths of length up to $n + 1$, which is best possible, and this appears here with his kind permission.

LEMMA 3. *Let $C_n = \{0, 1\}^n$, $S = \{y_1, \ldots, y_n\} \subseteq C_n$ and $x \in C_n - S$. There exist paths $D_1, \ldots, D_n$ from x to $y_1, \ldots, y_n$, so that for $i \neq j$. D_i and D_j have only the node x in common, and length$(D_i) \le n + 1$, $1 \le i \le n$.*

PROOF. The unit vectors of C_n are, by definition, the vectors e_i, $1 \le i \le n$, such that $e_i[j] = \delta_{ij}$. Denote by U_n the set of all unit vectors. The lemma clearly follows from the following claim.

Let $U \subseteq U_n$, $H \subseteq C_n$, $|H| = |U| = k$, $U \cap H = \emptyset$. Then there exist k vertex-disjoint paths $F_1, \ldots, F_k$, connecting the nodes in U to the nodes in H. If $z \in H \cap U_n$, then the path F_i connecting it to a node $e_i \in U$ has just these two nodes in U_n. If $z \in H - U_n$, then the connecting path has just the node e_i in U_n. Finally, length $(F_i) \le n$, $1 \le i \le k$.

The proof of this claim is by induction on n. The case $n = 2$ is dealt with by inspection. Assume validity for C_n. Let $U \subseteq U_{n+1}$, $H \subseteq C_{n+1}$, $|H| = |U| = k \le n + 1$, $U \cap H = \emptyset$. The cube C_{n+1} can be viewed as the union $C_{n+1} = C_n^0 \cup C_n^1$ where $C_n^0 = C_n \times \{0\}$, $C_n^1 = C_n \times \{1\}$. Thus to $x \in C_n$ there correspond $(x, 0)$, $(x, 1) \in C_{n+1}$, and we shall use the notations p, q, to denote the functions $p((x, 0)) = (x, 1)$, $q((x, 1)) = (x, 0)$.

We must consider several cases.

Case 1. Assume $H \subseteq C_n^0$, $e_{n+1} = (0, \ldots, 0, 1) \notin U$, then there is nothing to prove. If $e_{n+1} \in U$, then connect by the induction hypothesis the nodes in $U - \{e_{n+1}\}$ by $k - 1$ paths to $k - 1$ nodes in H. If one of the paths to some $z \in H$ passes through the remaining point w, then remove the path segment from w to z. In either case we now have $k - 1$ paths from $U - \{e_{n+1}\}$ to $H - \{u\}$, where u is z or w. Connect e_{n+1} by a shortest path F in C_n^1 to $p(u)$. The path (F, u) is the kth required path.

Case 2. Assume $H \subseteq C_n^1$. This is dealt with in a similar manner by connecting e_{n+1} in C_n^1 to a nearest node $z \in H$, and connecting the nodes in $U - \{e_{n+1}\}$ to the nodes in $q(H - \{z\})$. If it happens that for $y \in H - \{z\}$, $q(z) = e_i \in U$, then take (e_i, y) as the required path and remove y from H and e_i from U. The induction hypothesis allows the connecting paths in C_n^0 to avoid e_i.

Case 3. $H = H_0 \cup H_1$, $H_0 \subseteq C_n^0$, $H_1 \subseteq C_n^1$, $e_{n+1} \in U$. Let $|H_1| = m \leq n$. For $m - 1$ unit vectors (of C_{n+1}) in U, say $e_1, \ldots, e_{m-1}$, consider $p(e_1), \ldots, p(e_{m-1}) \in C_n^1$. If any $p(e_i) \in H_1$, take $(e_i, p(e_i))$ as the required path and remove e_i from U and $p(e_i)$ from H. Thus without loss of generality $p(e_i) \notin H_1$, $1 \leq i \leq m - 1$.

Viewing e_{n+1} as the origin in C_n^1, the vectors $p(e_1), \ldots, p(e_n)$ are the unit vectors of C_n^1. Recall that $m \leq n$ so that $m - 1 < n$. Thus if $p(e_m) \notin H_1$, then $\{p(e_1), \ldots, p(e_{m-1}), p(e_m)\}$ and H_1 satisfy the induction hypothesis in C_n^1 and there are m nonintersecting paths of length at most n from the nodes $p(e_i)$, $1 \leq i \leq m$, to the nodes in H_1. Augmenting the path G starting at $p(e_m)$ to (e_{n+1}, G) and each path G_i starting at $p(e_i)$, $1 \leq i \leq m - 1$ to (e_i, G_i), provides disjoint paths from nodes in U to all nodes in H_1, and all paths are of length at most $n + 1$.

We must still connect from $U - \{e_1, \ldots, e_{m-1}, e_{n+1}\}$ to H_0. But this is possible by the induction hypothesis for C_n^0, and the paths do not pass through any of the unit vectors in $U_{n+1} \cap C_n^0$ except those in $U - \{e_1, \ldots, e_{m-1}, e_{n+1}\}$.

Case 4. Similar to Case 3, but $e_{n+1} \notin U$. The proof is similar.

The above proof provides a very rapid recursive algorithm for constructing the paths in Lemma 3.

5. *Simplified Route Computation*

The Routing Algorithm of Theorem 1 requires the computation of the paths $D_i(x)$, $E_i(x)$ by the algorithm of Lemma 3. Although the algorithm is fast, the computation time for $2n$ paths per packet Px may be deemed expensive. We therefore propose another routing method in which the pieces emanating from a node are first distributed to fixed locations that are mutually far apart, and then dispatched to random intermediate destinations.

A matrix $H = [d_{ij}]$ is Hadamand if $d_{ij} = \pm 1$ and

$$i \neq j \quad \text{implies} \quad d_{i1}d_{j1} + \cdots + d_{in}d_{jn} = 0. \qquad (12)$$

For the sake of simplicity we assume that the dimension n of the cube is itself a power of 2 or that $n - 1$ is a prime, $n \equiv 0 \bmod 4$ (e.g., $n = 12, 24$). For such n, and many others there exist $n \times n$ Hadamand matrices [3] H_n that are very rapidly calculable.

Condition (12) implies that two different rows in H_n differ in exactly $n/2$ locations. Define $v_i \in C_n$, $1 \leq i \leq n$ by $v_i[j] = 1$ if $d_{ij} = 1$, $v_i[j] = 0$ if $d_{ij} = -1$. Then $d(v_i, v_j) = n/2$ for $i \neq j$, where $d(v, u)$ is the Hamming distance. The author has directly constructed systems of vectors $v_1, \ldots, v_n$ with this mutual distance property. The connection to Hadamand matrices was pointed out by D. Sleator (private communication).

As stated before, for a given n of the appropriate form, H_n and hence the v_i, $1 \leq i \leq n$, can be rapidly computed. Alternatively, we can precompute and store these vectors. By renumbering we can arrange it so that $v_i[i] = 1$, $1 \leq i \leq n$.

We can now outline a routing algorithm for implementing a permutation $\pi \in S_N$.

```
cobegin for $x \in C_n$:
(1) Split $Px$ into $n$ pieces $Px1, \ldots, Pxn$;
(2) Choose randomly $n$ nodes $R_1(x), \ldots, R_n(x)$;
(3) Send each piece $Pxi$ from $x$ to $x + v_i$;
(4) Send each piece $Pxi$ from $x + v_i$ to $R_i(x)$ along shortest path;
(5) Send each piece $Pxi$ from $R_i(x)$ to $\pi(x) + v_i$ along shortest path;
(6) Send each $Pxi$ from $\pi(x) + v_i$ to $\pi(x)$.
coend.
```

THEOREM 4. *With the above algorithm and the same assumption concerning buffer size as in Theorem* 1, *the probability that all packets will arrive at their destinations is at least* $1 - N^{-4}$. *For an appropriate* $0 < c$, *if up to* N/n *links randomly fail, the probability that no packet will be lost through link failure is at least* $1 - n^{-c \cdot n}$.

The proof of the first statement is a simple variant of the proof of Theorem 1. The proof for the fault-tolerance statement proceeds by showing that the randomly selected portions of the paths for pieces *Pxi* of the same packet *Px* are, with high probability, almost edge-disjoint. Details of the argument and additional results will be given elsewhere.

The above routing method avoids costly route computations because the routing tickets for phases 3 and 6 can be precomputed (or else computed on the fly), and the tickets for phase 4 and 5 involve just random bits, which again may be computed on the fly. Yuh-Dauh Lyuu has another very simple routing method achieving the same results as in Theorem 4.

6. *Conclusions and Further Directions*

The IDA has numerous potential applications to secure and fault-tolerant storage and transmission of information. A beneficial side effect of using IDA is improved load balancing in storage and transmission. All of these claimed benefits of IDA are quantitatively analyzed and demonstrated in the case of packet switching on the n-cube.

In practical applications to the latter problem, one need not strictly adhere to the routing algorithms of Sections 4 and 5. One can elect for a given dimension, say $d = 16$, to split packets Px into a number n of pieces smaller than d, say $n = 6$ with $m = 4$ sufficing for reconstruction. It is then possible to test by experimentation what assumption concerning buffer sizes and faults lead to acceptable levels of performance and fault-tolerance. In fact, the (n, m) parameters of IDA can be dynamically tuned to message loads.

Returning to the general algorithm, we conjecture that Theorem 1, strong as it is, does not reflect the full power of applying IDA. We conjecture that the probability of all packets reaching their destinations in a transmission round is at least $1 - N^{-c \cdot n}$, for an appropriate not very small $0 < c$.

An obvious extension of this work is to consider network topologies other than the n-cube.

REFERENCES

1. ASMUTH, C. A., BLAKLEY, G. R. Pooling splitting and restituting information to overcome total failure of some channels of communication. In *Proceedings of the 1982 Symposium on Security and Privacy*. IEEE Society, New York, 1982, pp. 156–169.

2. BERLEKAMP, E. R. Algebraic coding theory. McGraw-Hill, New York, 1968.
3. HALL, M. *Combinatorial Theory.* Wiley, New York, 1980.
4. HILLIS, W. D. *The Connection Machine.* MIT Press, Cambridge, Mass., 1985.
5. MIRSKY, L. *An Introduction to Linear Algebra.* Dover, New York, 1982.
6. PIPPENGER, N. Parallel communication with limited buffers. In *Proceedings of the IEEE 25th Symposium on Foundations of Computer Science.* IEEE, New York, 1984, pp. 127–136.
7. RABIN, M. O. Probabilistic algorithms in finite fields. *SIAM J. Comput. 9*, 1980, 273–280.
8. RABIN, M. O. Fingerprinting by random polynomials. Tech. Rep. TR-15-81. Center for Research in Computing Technology. Harvard Univ., Cambridge, Mass., 1981.
9. RAGHAVAN, P. Probabilistic construction of deterministic algorithms: Approximating packing integer programs. In *Proceedings of the IEEE 27th Symposium on Foundations of Computer Science.* IEEE, New York, 1986, pp. 10–18.
10. SEITZ, C. L. The cosmic cube. *Commun. ACM 28*, 1 (Jan. 1985), 22–33.
11. SHAMIR, A. How to share a secret. *Commun. ACM 22*, 11 (Nov. 1979), 612–613.
12. SIEGEL, H. J. A model of SIMD machines and a comparison of various interconnection networks. *IEEE Trans. Comput. 28*, 12 (1979), 907–917.
13. VALIANT, L. G. A scheme for fast parallel communication. *SIAM J. Comput. 11*, 2 (1982), 350–361.

RECEIVED MAY 1987; REVISED JUNE 1988; ACCEPTED AUGUST 1988