Yale University
Department of Computer Science

Constrained Randomization For Parallel Communication

Abhiram G. Ranade

YALEU/DCS/TR-511
January 1987

This work has in part been supported by the Office of Naval Research under contracts N00014-84-K-0043 and N00014-86-K-0564. Approved for public release: distribution is unlimited.

Constrained Randomization For Parallel Communication *

Abhiram G. Ranade

Yale University
Department of Computer Science
New Haven, Connecticut

Abstract: Several algorithms for parallel communication include random reorganization of messages as an intermediate step. This paper presents a technique by which this step can be carried out in exactly $\log N$ message cycles, where N is the number of processors, for a large class of realistic networks. Although the randomization accomplished is not total, it is shown to be sufficient for some of the parallel communication algorithms.

1 Introduction

Randomization was proposed by Valiant and others [1,6,10,11] as a solution to the problem of parallel communication on a network computer. Their methods consist of a randomization phase in which the messages to be delivered are sent to independent randomly chosen destination, followed by a routing phase in which the messages are forwarded to the correct destinations. For realistic network computers (i.e. with a realistic number of processors, each processor having only a few communication links) consisting of N processors, randomized communication accomplishes message delivery in $O(\log N)$ time, as opposed to $O(\log^2 N)$ time or larger for practical deterministic algorithms [2,3,8].

The subject of this paper is the randomization phase. A technique called *Constrained Randomization* is proposed, using which the randomization phase can be completed in deterministic time, while still guaranteeing message delivery in the routing phase in $O(\log N)$ time with high probability. Using constrained randomization, messages are not sent to totally randomly chosen intermediate destinations, but instead are permuted, the permutation being randomly chosen

*This work has in part been supported by the Office of Naval Research under contracts N00014-84-K-0043 and N00014-86-K-0564.

in a distributed manner from a large set $\mathcal{P}$ of permutations. In particular, the randomization is performed using an Omega network and $\mathcal{P}$ is the set of permutations that can be performed in one pass through the Omega network. It will be seen that constrained randomization has two immediate advantages. First, the randomization is accomplished faster viz. in exactly $x - 1 + \log N$ message cycles for x message sets. As opposed to this, total randomization can be acheived in $k(x - 1 + \log N)$ message cycles only with high probability, with $k \approx 8$, typically. Secondly, constrained randomization uses half as many random bits as are required for complete randomization. It is felt that constrained randomization may be useful for problems besides data movement.

This paper is organized as follows. Sections 2 and 3 define the network configuration and the communication model. Sections 4 and 5 analyze the problem of routing a single message permutation. Sections 6 through 8 consider the problem when each source has multiple messages to send. The discussion uses a Benes network formed using an Omega and an inverse Omega network. But it is applicable to other networks like the hypercube, cube connected cycles etc. because of well known transformations [9,12].

2 Configuration

The architecture being considered here consists of N message sources and N message sinks connected together by two interconnection networks. The first interconnection network, called the *Randomizer* connects the N message sources to N nodes called *intermediate destinations*. The second network, called the *Router* connects the N intermediate destinations to the N message sinks.

The Randomizer sends the messages originating at the message sources to random intermediate destinations. This is accomplished in exactly $\log N$ time, and no queueing is required within the Randomizer. The Router is a parallel communication system (described later), and is used to route the messages from the intermediate destinations to the final destinations.

In this paper, the function log is used to denote the logarithm of a number to the base 2. Also, the bits in an n bit binary number are numbered 1 through n from the most significant bit through the least significant.

2.1 Randomizer

The Randomizer ($\mathcal{RND}$) is an Omega network [5,12].

An n stage Omega network has $N = 2^n$ inputs and outputs. The stages are numbered 1 through n starting from the input side. Each stage contains $N/2$ switches. Each switch has two inputs and two outputs, and a capability to generate a random bit every cycle. The switch can either route the messages it receives on its inputs onto corresponding outputs, or exchange them before routing them. The random bit is used to decide whether or not to exchange the messages. These switches are numbered 0 through $N/2-1$, and stage inputs/outputs $p_1..p_{n-1}0$ and $p_1..p_{n-1}1$ form the inputs/outputs to switch $p_1..p_{n-1}$. Output $p_1..p_n$ of stage i is connected to input $p_2..p_np_1$ of stage $i+1$. Switches in stage i are called *i-switches.* Network output j is taken from output j of stage n. Network input $p_1..p_n$ is connected to input $p_2..p_np_1$ of stage 1.

If s_a and s_b are switches, then $stage(s_a)$ will denote the stage of s_a. If there is a path between s_a and s_b and $stage(s_a) < stage(s_b)$ then $s_a < s_b$. The other operators $>, \leq, \geq, =$ are similarly defined. Open and closed switch intervals are also defined e.g. $[s_a, s_b) = \{s \mid s_a \leq s < s_b\}$.

The following proposition describes how a message can travel from input $a_1..a_n$ of an Omega network to output $b_1..b_n$. The proof can be found in [5].

Proposition 1 *There exists a unique path from input $a_1..a_n$ of an Omega network to output $b_1..b_n$, and this path passes through input $a_{i+1}..a_nb_1..b_{i-1}a_i$ and output $a_{i+1}..a_nb_1..b_i$ of stage i.*

2.2 Router

The Router ($\mathcal{RTE}$) is an inverse Omega network. An inverse Omega network is an Omega network with inputs and outputs exchanged. The stages of an inverse Omega network will be assumed to be numbered 1 through n starting from the output side. Again, the switches in stage i will be called i-switches. As for the Randomizer, the following proposition describes how messages travel from the input $a_1..a_n$ to the output $b_1..b_n$.

Proposition 2 *There exists a unique path from input $a_1..a_n$ of an inverse Omega network to output $b_1..b_n$, and this path passes through input $b_{i+1}..b_na_1..a_i$ and output $b_{i+1}..b_na_1..a_{i-1}b_i$ of stage i.*

2.3 The Communication Model

The Router is a parallel communication system as described by Valiant[11]. A parallel communication system (PCS) has the following features:

1. There is a directed graph $G = (V, E)$ where V is a set of nodes, and E the set of edges between them.

2. There are a total of T messages in the system at any time. Each message carries with it a *ticket* for book-keeping. Initially the ticket contains information about the name of the message, its destination address, and possibly a total or partial specification of the path. The ticket is used to route the message towards its destination, and the routing of a message only depends on the contents of its ticket. In particular the routing does not depend upon the presence of other messages at the node. During routing, the ticket of a message may be altered. Again, these alterations depend only upon the original contents.

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

4. At every node there is one queue for each outgoing edge. At the start of each unit interval, each queue contains a set of messages, and the *Queueing Discipline* determines which one is to be at the head of the queue, provided the queue is non-empty. During the unit interval, the message at the head of every non-empty queue is transmitted along the associated directed edge to the neighbouring node.

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

Some definitions follow. The *delay* of a message is the total time it waits unserved in the queues. The *route* is the path it follows in G. A message A is said to *delay* a message B at time t if at time t A and B are present in the same queue at some node and A is chosen for transmission. Two messages are said to *collide* if their routes share an edge (irrespective of when they pass through the shared edge). A *run* R of a Parallel Communication Scheme is a specification of the node at which a message is at a given instant, i.e $R : \Lambda \times I \rightarrow V$ where Λ is the set of messages, and I is the set $\{0, 1, 2..\}$ (time). A scheme is *non-repeating* if whenever two packets take paths $e_1..e_r$ and $f_1..f_s$ (with $e_i, f_i \in E$) in which $e_j = f_k$ and $e_l = f_m$ $(j > l)$ it is the case that $j - l = k - m$ and $e_p = f_{p+k-j}$ for $l \leq p \leq j$.

We require the following result, which holds for all queueing disciplines in non-repeating communication schemes. The proof can be found in [11].

Proposition 3 *If a message has a delay of d, then there are at least d other messages which collide with it.*

2.4 The Router As A PCS

For the Router, G is the graph of the inverse Omega network. The routing algorithm is fairly simple: at each switch the appropriate bit of the message destination is used to place the message onto the correct output queue. The ticket for each message is simply its final destination. Finally, because there is a unique path between any source and any destination, it follows that the scheme is non-repeating.

3 Benes Network Characteristics

This section derives some results about the structure of the Benes Network.

Some notation is first defined. Capital Roman letters are used to identify messages. If A is a message, then A', A'', A''' denote the numerical address of the source, the intermediate and the final destination respectively. a'_i denotes the i^{th} bit of the binary representation of A' and so on. Thus, for example $A'' = a''_1 a''_2 .. a''_n$ in binary. If x is a single bit, then $\overline{x}$ denotes the complement of x.

Lemma 1 *If the paths of messages A and B share an i-switch without sharing a link in $\mathcal{RND}$ then $a''_i = \overline{b}''_i$ and $a'_i = \overline{b}'_i$.*

Proof: *In $\mathcal{RND}$ the messages A and B enter the same switch through separate inputs. Thus $a'_{i+1}..a'_n a''_1..a''_{i-1} a'_i = b'_{i+1}..b'_n b''_1..b''_{i-1} \overline{b}'_i$. Thus $a'_i = \overline{b}'_i$. Similarly, they leave the switch at different outputs. Thus $a'_{i+1}..a'_n a''_1..a''_i = b'_{i+1}..b'_n b''_1..b''_{i-1} \overline{b}''_i$. Thus $a''_i = \overline{b}''_i$*

Lemma 2 *If the paths of messages A and B separate at an i-switch in $\mathcal{RTE}$ after sharing at least one edge, then $a'''_i..a'''_n a''_1..a''_i = \overline{b}'''_i b'''_{i+1}..b'''_n b''_1..b''_i$*

Proof: *The two messages share at least an edge, hence they must enter at the same input of the i-switch. Thus $a'''_{i+1}..a'''_n a''_1..a''_i = b'''_{i+1}..b'''_n b''_1..b''_i$. Since their paths separate at this switch, these must be the two outputs of the same switch, thus: $a'''_{i+1}..a'''_n a''_1..a''_{i-1} a'''_i = b'''_{i+1}..b'''_n b''_1..b''_{i-1} \overline{b}'''_i$. Hence the result follows.*

Theorem 1 *In $\mathcal{RTE}$, if the paths of distinct messages A and B leave the path of a message M at an i-switch and a j-switch respectively after sharing at least one edge (separately, perhaps), with $i \leq j$, then in $\mathcal{RND}$, the path of A upto an i-switch is switch disjoint from the path of B upto a j-switch.*

Proof: *By lemma 2 $a'''_i..a'''_n a''_1..a''_i = \overline{m}'''_i m'''_{i+1}..m'''_n m''_1..m''_i$ and $b'''_j..b'''_n b''_1..b''_j = \overline{m}'''_j m'''_{j+1}..m'''_n m''_1..m''_j$ Thus $a''_1..a''_i = m''_1..m''_i = b''_1..b''_i$ because $i \leq j$. Hence for any $k \leq i$ $a''_k = b''_k$. Let, in fact the two paths in $\mathcal{RND}$ share a k-switch, with $k \leq i$. In $\mathcal{RND}$, by construction the messages cannot share links. Thus lemma 1 is applicable and thus $a''_k = \overline{b}''_k$, contradiction.*

4 Permutation Routing

Suppose each of the N message sources has one message initially, destined for one of the message sinks such that each of the N message sinks receives exactly one message. It will be shown that these messages can be delivered in $O(\log N)$ time with high probability.

4.1 Proof Technique

The proof technique is adapted from [7] and is based on the concept of Kolmogorov-complexity [4].

Let T be the class of one dimensional Turing machines with tape alphabet $\{0, 1, B\}$. Let U be a Universal Turing machine in T. For $w_1, w_2 \in \{0,1\}^*$ define the Kolmogorov-Complexity by:

> $K(w_1 \mid w_2) \equiv$ The length of the shortest 0/1 string ('program, description of w_1') p, such that U with input pBw_2 computes w_1 and stops. $K(w) \equiv K(w \mid \text{emptystring})$.

A simple counting argument can be used to prove the following proposition, stated here from [7]. This result says that strings that have short descriptions are unlikely to be generated in random trials.

Proposition 4 *Let $w_2 \in \{0,1\}^*$ be fixed and chose $w_1 \in \{0,1\}^n$ by tossing a fair coin n times. Then for all c:*

$$\textit{probability}(\{w_1 \mid K(w_1 \mid w_2) < n - c\}) \leq 2^{-c}$$

4.2 Kolmogorov Complexity of a Run

Given a deterministic routing algorithm, the only variation in runs is that caused by the random bits selected in $\mathcal{RND}$. The number of bits chosen at random is $(N \log N)/2$. Thus there are $2^{(N \log N)/2}$ different runs. Each different run is completely identified by the random bits chosen.

Let ξ be an ordering of the switches in $\mathcal{RND}$, with $\xi(i)$ representing the *ith* switch as per the ordering. Let π represent the message permutation encoded in some form. Given π, a run R of the parallel communication system can be completely identified by a bit string w_R where the *ith* bit of w_R is the bit generated at switch $\xi(i)$.

Definition 1 *The Kolmogorov-Complexity of a run R given π is defined as $C(R) \equiv K(w_R \mid \pi)$.*

It will be shown in the following section that runs having large delays for a message permutation π have descriptions of length less than $|w_R| = 2^{N \log N/2}$ and thus, low probability.

5 Description Of A Run

The description of a run consists of the following components:

1. Description of the intermediate and the final destinations of a message M suffering a maximum delay.

2. Description of messages that collide with M. This description is in n parts, the i^{th} part being the description of the destinations of the set of messages leaving the path of M at an i-switch.

3. Serial specification of all the random bits which cannot be deduced from the above.

5.1 Describing the message with the maximum delay

Specification of the intermediate and final destinations of M requires n bits each. The source of M' can be found using π. Thus, knowing M' and M'', it is possible to construct the path of M in $\mathcal{RND}$. This reveals n random bits, one for every switch on the path.

5.2 Describing Colliding messages

Let C_i messages leave the path of M at an i-switch in $\mathcal{RTE}$. By lemma 2 $a_i'''..a_n'''a_1''..a_i'' = \overline{m_i'''}m_{i+1}'''..m_n'''m_1''..m_i''$ for every such message A, i.e., all the C_i messages must have destinations of the form $**..*m_i'''..m_n'''$. But each message has a unique destination. Thus there are $\binom{2^{i-1}}{C_i}$ possibilities. Thus naming one of these requires $\log\binom{2^{i-1}}{C_i}$ bits. The number of bits required is variable, and thus there is a parsing problem. This can be solved by prefixing the above bit string with a parsable string that encodes C_i. Since i is known, it is possible to compute $\log\binom{2^{i-1}}{C_i}$. C_i can be represented in a parsable form using $2\log C_i$ bits as follows. The first component of the description consists of a string of 1's of length $\log C_i$. This is followed by a zero, and then by the representation of C_i in binary which requires another $\log C_i$ bits. The complete description of the set for a given i is as follows: string of $\log C_i$ 1's terminated by a 0; value of C_i in binary, which requires $\log C_i$ bits; and the description of the C_i messages, which requires $\log\binom{2^{i-1}}{C_i}$ bits. Thus a total of $\log\binom{2^{i-1}}{C_i} + 1 + 2\log C_i$ bits are required.

The above encoding identifies the set of messages leaving the path of M at an i-switch by their final destinations. Given the permutation π, it is possible to find their sources. The most significant i bits of the intermediate destination are known as well, and thus this allows one to construct the path of every message upto and including an i-switch in $\mathcal{RND}$. However by theorem 1, all these paths, for all i are switch disjoint. Specifying a path in $\mathcal{RND}$ reveals the state of all the switches through which it passes. Since each path upto the output of an i-switch passes through i switches, $C_i i$ switch settings are encoded in this description. The gain in description is:

$$C_i i - \log\binom{2^{i-1}}{C_i} - 1 - 2\log C_i \geq C_i + C_i \log\frac{C_i}{e} - 2\log C_i - 1$$

Using $\binom{n}{k} \leq (\frac{ne}{k})^k$. This can be summed over different values of i. Let C represent the total number of touching messages and so $C = \sum_{i=1}^{n} C_i$. The total gain can then be estimated using the concavity of the functions $x \log x$ and $-\log x$. Thus the total gain is at least:

$$\sum_{i=1}^{n}(C_i + C_i \log \frac{C_i}{e} - 2\log C_i - 1) \geq C + C \log \frac{C}{ne} - 2n \log \frac{C}{n} - n$$

5.3 Describing the rest

The switches in $\mathcal{RND}$ are ordered canonically, and those that are included in the description above are marked off. The random bits generated by all the unmarked switches are described in the canonical order. This description therefore requires exactly as many bits as there are unmarked switches, and thus there is no gain or loss in the description.

5.4 Conclusion

Thus the total gain in the representation can be estimated by adding up the separate gains and losses. Thus from sections 5.1 and 5.2 the total gain is at least:

$$C + C \log \frac{C}{ne} - 2n \log \frac{C}{n} - 2n > Kn$$

for $C = Kn$ with $K \geq 8$. Thus the probability that a run does not complete in time $(K+1)\log N$ is:

$$< 2^{-Kn} = N^{-K}$$

6 Pipelining Communication

This section examines the problem of routing multiple permutations. Each of the N message sources is assumed to have x messages initially. The messages at each node have priorities 0 through $x-1$. It is assumed that messages of a given priority form a permutation, i.e. each of the N message sinks receives exactly one message of each priority. The priorities are used by the queueing discipline to determine which message to transmit at every node. Specifically, a message with the highest priority is chosen for transmission. Between messages of the same priority, the choice is arbitrary. Such queueing disciplines will be called *priority-based queueing disciplines.* The Randomizer works in deterministic time as before, and its operation is pipelined. Thus, the x sets of messages arrives at the intermediate destinations in $n + x - 1$ time units. The total number of random bits generated by the Randomizer is $xN \log N/2$. It will be assumed that the Router starts transmission only after all the messages arrive at the intermediate destinations. This assumption somewhat simplifies the argument that follows, but it should be clear that this extra delay is unnecessary.

It will be shown that under any priority-based queueing discipline, all the messages will be delivered in time proportional to $n+x-1$ with very high probability. The proof technique used is similar to that used earlier, i.e. the runs of the parallel communication system will be encoded and runs having large delays will be shown to have compact encodings, and thus low probabilities. Given the destinations Π for each of the Nx messages, and the queueing discipline at each switch of the Router, the $xN \log N/2$ bits generated in the Randomizer uniquely identify a run, i.e. a run R can be completely identified by a bit string v_R which specifies the random bits generated. Thus the encoding problem may be formally stated as follows.

Definition 2 *The Kolmogorov-Complexity of a run R given Π is defined as $C(R) \equiv K(v_R \mid \Pi)$.*

It will be shown that runs having large delays can be described using considerably fewer bits than $|v_R| = xN \log N/2$, and thus are improbable.

6.1 Monotonic Runs

As mentioned above, the queueing discipline at each switch is only required to give preference to messages according to priority, it may treat messages with the same priority arbitrarily. The following theorem shows that it is sufficient to consider queueing disciplines which are somewhat restricted, for purposes of encoding runs. In particular, it is shown, that given a run R which uses an arbitrary priority-based queueing discipline, it is possible to construct a run R' which is *monotonic*. R' rather than R is used to encode the bits generated in the Randomizer.

Definition 3 *A run R is said to be monotonic if for all messages A and B the following is satisfied: If a message A ever delays another message B, then message B never delays A.*

Definition 4 *Consider a run R of $\mathcal{RND}$ in which messages A and B delay each other. Let A delay B at switch and time (s_1, t_1) and B delay A at (s_2, t_2). Let $s_1 < s_2$, i.e. s_1 is towards the output side. Because the PCS being considered is non-repeating, A and B have the same route between s_1 and s_2. Let neither A nor B delay the other between s_1 and s_2. Then the messages A and B are said to constitute an inconsistency $(A, B, s1, t1)$ at $(s1, t1)$. Given inconsistencies $\alpha = (A, B, s, t)$ and $\alpha' = (A', B', s', t')$ $\alpha \leq \alpha'$ if $stage(s) < stage(s')$ or $stage(s) = stage(s')$ and $t \leq t'$. If $\alpha \leq \alpha'$ for all α' then α is said to be a least inconsistency.*

Theorem 2 *Given a run R and a priority-based queueing discipline QD for $\mathcal{RTE}$ it is possible to construct a monotonic run R' and an associated priority-based queueing discipline QD' such that:*

1. *The initial message sets for the two runs are identical.*

2. *The delay suffered by any message from the intermediate source to the destination is the same for both runs.*

Proof: *If R contains no inconsistencies, then clearly R is monotonic. If R does contain inconsistencies, then it must contain a least inconsistency α. It will be shown that it is possible to remove α without introducing inconsistencies $\beta \leq \alpha$. Since a run cannot contain arbitrarily large inconsistencies, this process must result in a removal of all inconsistencies.*

Let $\alpha = (A, B, s_1, t_1)$ be a least inconsistency, with B delaying A at (s_2, t_2). Consider the run R_1 obtained by exchanging A and B between s_2 and s_1.[1] *Assume that an inconsistency $\beta \leq \alpha$ is introduced in R_1 at (s_i, t_i) because of the above exchange. This must involve either A or B. If this involves A, then it must be caused because a message C delays A at (s_i, t_i), and A delays C at some (s_j, t_j) where $s_1 < s_j \leq s_2$, and w.l.o.g. A and C do not delay each other in (s_i, s_j). Note that it is easy to show that $s_i < s_1$.*

Thus in R, B delays C at (s_j, t_j) and B and C do not delay each other in $[s_1, s_j]$. In R, at the output of s_j, assume A is ahead of C. But A is known to be behind B, and thus because B delays C, A delays C as well. But in this case, the inconsistency $(C, A, s_i, t_i) \leq \alpha$ in R, proving that α is not a least inconsistency in R.

Thus at the output of s_j, C must be ahead of A. Because B remains ahead of C in $[s_1, s_j]$, and A is ahead of B at s_1, A must also be ahead of C. Thus A must delay C in $[s_1, s_j)$. Again, the inconsistency $(C, A, s_i, t_i) \leq \alpha$.

A similar contradiction is obtained if the inconsistency involves B rather than A in R_1.

Definition 5 *A delaying set: $DS(M, s)$ for a message M at switch s is the set of messages which delay M at s, or delay any message which delays M etc.*

Let $d(M, s)$ represent the delay suffered by a message M at switch s. Let $D(M, s_i) = \sum_{s=s_n}^{s_i} d(M, s)$, where s_i is the switch on the path of M such that $stage(s_i) = i$, and the sum is taken along the path of M.

Theorem 3 *In $\mathcal{RTE}$ let the switches on the path of a message M be labelled s_1 through s_n from the output to the input. For a message M' in $DS(M, s_i)$ and $1 \leq j \leq i \leq n$*

$$\left| \bigcup_{s=s_i}^{s_j} DS(M, s) \right| + D(M', s_{i+1}) \geq \sum_{s=s_n}^{s_j} d(M, s)$$

Proof: *The proof is by induction on j.*

M' enters a time $D(M, s_{i+1}) - D(M', s_{i+1})$ before M enters, and $D(M, s_i) - D(M', s_{i+1})$ before M leaves. Because M' delays M there must be at least

[1]The associated queueing discipline QD_1 is priority-based because A and B are guaranteed to have the same priority.

$D(M, s_i) - D(M', s_{i+1})$ other messages that delay M. But these messages are in $DS(M, s_i)$. Thus $|DS(M, s_i)| \geq D(M, s_i) - D(M', s_{i+1})$, establishing the base case. The induction hypothesis is:

$$\left|\bigcup_{s=s_i}^{s_{j+1}} DS(M,s)\right| + D(M', si+1) \geq \sum_{s=s_n}^{s_{j+1}} d(M,s)$$

Let $\left|\bigcup_{s=s_n}^{s_{j+1}} DS(M,s) \bigcap DS(M,s_j)\right| = \alpha$. Then by monotonicity all these α messages which indirectly delay M before s_j must enter s_j before M, i.e., the first of these must arrive at least a time α before M arrives. Thus,

$$|DS(M,s_j)| \geq \alpha + d(M,s_j)$$

Adding this to the induction hypothesis, the result follows, using:

$$\alpha = \left|\bigcup_{s=s_n}^{s_{j+1}} DS(M,s) \bigcap DS(M,s_j)\right| \geq \left|\bigcup_{s=s_i}^{s_{j+1}} DS(M,s) \bigcap DS(M,s_j)\right|$$

Corollary 1 *Let the switches on the path of a message M be labelled s_1 through s_n from the output to the input. Then for $1 \leq i \leq n$*

$$\left|\bigcup_{s=s_n}^{s_i} DS(M,s)\right| \geq D(M,s_i)$$

Proof: Obvious.

7 Construction Of A Distinguished Path

As mentioned above, it is only necessary to consider monotonic runs. The description of a run can be made compact because every monotonic run has at least one *distinguished* path which touches a large number of messages in a simple manner. This section describes how such a path may be found given a run R.

A distinguished path goes from an input of the Router to an output, and is made up of paths of several messages. If x is the number of priorities, then the set Q of switches on this path can be partitioned into x sets $Q_0..Q_{x-1}$ having the following characteristics. Each Q_i is either empty or contains switches adjacent on the distinguished path. Each non-empty Q_i corresponds to a part of the path of a message M_i of priority i, i.e. M_i passes through every switch in Q_i. Finally, Q_i are encountered in ascending order along the distinguished path starting from the output side i.e. the switches in Q_i are closer to the output side than Q_j if $i < j$. The following paragraph describes how a distinguished path can be constructed starting from the output side of the Router.

Let M be a message with the maximum delay, with p as its priority. Then $M_p = M$, and all $Q_0..Q_{p-1}$ are null sets. Let s_p be the final switch on the path of M_p. In general, given M_i and s_i, Q_i is constructed as follows. Let s_i' be the first switch on the path of M_i starting from s_i and moving toward the input, such that $DS(M_i, s_i')$ contains a message with priority higher than i. Then the $Q_i = [s_i, s_i']$. Let M_j be a highest priority message in $DS(M_i, s_i')$ that is not delayed by any other message at s_i'. Let s_j be the first switch on the path of M_j starting from s_i' and moving towards the input. Then, given s_j and M_j, set Q_j is constructed as above, and all sets $Q_{i+1}..Q_{j-1}$ are null sets. In this manner all $Q_0..Q_{x-1}$ are constructed.

8 Description Of A Run

For a given Q_i and M_i, the messages in $\bigcup_{s\in Q_i} DS(M_i, s)$ can be described compactly. The description consists of $|Q_i| + j - i$ components, assuming the first non null segment following Q_i is Q_j. The first $|Q_i|$ components describe the messages of priority i that delay M_i at any of the $|Q_i|$ switches in Q_i. If a message delays M_i at more than one switch, it is included in only one of the corresponding components. M_i can also be delayed by messages of higher priority. But by construction, these messages can delay M_i only at s_i'. Further, the highest priority message that delays M_i has priority j. Thus these messages can be described in $j - i$ groups, each group consisting of messages of a given priority. Thus the total number of groups is $\sum_k |Q_k| + x - 1 = n + x - 1$ (There could be some null groups). Further, these groups are encountered in order of increasing priority as one moves up the distinguished path towards the input side.

Because of monotonicity, the messages associated with Q_k viz. $\bigcup_{s\in Q_k} DS(M_k, s)$ are disjoint for different values of k. Thus, the total number of messages described:

$$\sum_k \left| \bigcup_{s\in Q_k} DS(M_k, s) \right|$$

If Q_i and Q_j are the last two segments i.e. the closest to the input side, then by corollary 1 and proposition 3 respectively:

$$\left| \bigcup_{s=s_i'}^{s=s_i} DS(M_i, s) \right| + \left| \bigcup_{s=s_j'}^{s=s_j} DS(M_j, s) \right| \geq \left| \bigcup_{s=s_i'}^{s=s_i} DS(M_i, s) \right| + D(M_j, s_j) \geq D(M_i, s_i)$$

Thus applying theorem 3 repeatedly:

$$\sum_k \left| \bigcup_{s\in Q_k} DS(M_k, s) \right| \geq D(M, s_p)$$

Thus the number of messages described is at least as large as the largest message delay. These messages are described in a manner similar to section 5. First, the distinguished path is described, followed by the messages touching the distinguished path. These are described in $n + x - 1$ groups, as discussed above. Finally all the random bits generated in $\mathcal{RND}$ which are not described earlier are explicitly listed.

8.1 Describing The Distinguished Path

This requires $2n$ bits, n to describe the intermediate source at which the path starts, and n to describe the destination. Let P be a fictitious message that travels along the distinguished path. Then the above specifies P'' and P'''.

8.2 Describing Touching Messages

This description is similar to that in section 5.2 with a few important differences. The description is in $n+x-1$ groups, each group describing messages of a given priority and touching the distinguished path at a given switch. Unlike section 5.2 the messages described in a group need not leave the path at the associated switch.

Each group is associated with a switch on the distinguished path and a priority. The groups are described in the same order in which the associated switches are encountered as one moves up the distinguished path towards the input, and groups associated with the same switch are described in the order of increasing priority. Notice that there is at least one group associated with every priority and every switch. Further if a switch s has g groups associated with it then these have g distinct and consecutive priorities. Finally, if s and t are consecutive switches on the distinguished path, then the highest priority of any group associated with s is the same as the least priority of any group associated with t. Thus if the number of groups associated with a switch is specified, then the priority and the switch associated with the j^{th} group $(0 \le j < n + x)$ can be computed. This can be done by specifying $n - 1$ 'boundaries' dividing the sequence of $n + x$ groups into n fragments, each fragment containing at least one group. This can be done in $\binom{n+x-1}{n-1}$ ways. Thus the specification requires $\log\binom{n+x-1}{n-1} = \log\binom{n+x-1}{x} \le x \log \frac{e(n+x-1)}{x}$ bits.

This paragraph describes how messages in group j $(0 \le j < n + x)$ can be described. Let the messages have priority k and let the associated switch be $s \in Q_k$. Let $stage(s) = i$. k and i can be computed given j as discussed in the previous paragraph. Let A be a message belonging to the group. A travels along the distinguished path at least one edge after s. Thus P and A leave s at the same output. Thus $a'''_{i+1}..a'''_n a''_1..a''_{i-1}a'''_i = p'''_{i+1}..p'''_n p''_1..p''_{i-1}p'''_i$. But P''' is known. Thus only the first $i - 1$ bits of A''' need be specified. Then given k, A''' and the message destinations Π it is possible to compute A' because there is a unique message of a given priority and destination. Note that the most significant $i-1$ bits of A'' are known because they are identical to those of P''. Thus it is possible to compute the path of A upto an $(i-1)$-switch in $\mathcal{RND}$. As in theorem 1 it can be shown that all such paths computed in $\mathcal{RND}$ are switch disjoint. As in section 5.2 the entire group may be encoded using $\log\binom{2^{i-1}}{C_j} + 1 + 2\log C_j$ bits, assuming the group contains C_j messages. The number of switch settings this specifies in $\mathcal{RND}$ is $(i-1)C_i$. Thus the gain in description for each group is

$$C_j(i-1) - \log\binom{2^{i-1}}{C_j} - 1 - 2\log C_j \geq C_j \log\frac{C_j}{e} - 2\log C_j - 1$$

Using $\binom{n}{k} \leq (\frac{ne}{k})^k$. This can be summed over the different groups using the concavity of the functions $x\log x$ and $-\log x$. Thus if $C = \sum_{j=0}^{j=n+x-1} C_j$, then the gain is at least:

$$\sum_{j=0}^{n+x-1} C_j \log\frac{C_j}{e} - 2\log C_j - 1 \geq C\log\frac{C}{e(n+x-1)} - 2(n+x-1)\log\frac{C}{n+x-1} - (n+x-1)$$

But at most $x\log\frac{e(n+x-1)}{x}$ bits are required to describe the number of groups associated with each switch. Thus the total gain is at least:

$$C\log\frac{C}{e(n+x-1)} - 2(n+x-1)\log\frac{C}{n+x-1} - (n+x-1) - x\log\frac{e(n+x-1)}{x}$$

8.3 Description Of The Rest

As in section 5.3 the random bits generated in $\mathcal{RND}$ are first ordered canonically, and then all the bits described above are marked off. The unmarked bits are described in the canonical order. Thus there is no gain or loss.

8.4 Conclusion

The total gain in the representation can be estimated by adding up the individual gains or losses. Thus from sections 8.1 and 8.2 the total gain is at least:

$$C\log\frac{C}{e(n+x-1)} - 2(n+x-1)\log\frac{C}{n+x-1} - (n+x-1) - x\log\frac{e(n+x-1)}{x} - 2n$$

$$\geq C\log\frac{C}{e(n+x-1)} - 2(n+x-1)\log\frac{C}{n+x-1} - 4(n+x-1)$$

Assuming $x > n$. Further, if $C = K(n+x-1)$ where $K \geq 9$ then the gain is at least:

$$C\log\frac{K}{e} - 2\frac{C}{K}\log 4K = C\log\frac{K}{e}\left(\frac{1}{4K}\right)^{\frac{2}{K}} \geq C/2 \geq Kn$$

Thus the probability of requiring more than $K(n+x-1)+n$ time to complete is less than:

$$2^{-Kn} = N^{-K}$$

Acknowledgements

I am very grateful to Lennart Johnsson for numerous discussions and for his careful reading of this manuscript, which lead to substantial improvements. I would also like to thank Sandeep Bhatt for many discussions.

References

[1] R. Aleliunas. Randomized parallel communication. In *PODC*, pages 60–72, 1982.

[2] K. Batcher. Sorting networks and their applications. In *AFIPS Spring Joint Comp. Conf.*, pages 307–314, 1968.

[3] Z. Galil and W. Paul. A practical general purpose parallel computer. In *Proceedings of STOC 81*, 1981.

[4] A. N. Kolmogorov. Three approaches to the quantitative definition of information. *Problems of Information Transmission*, 1-1:1–7, January-March 1965.

[5] K. Padmanabhan and D. H. Lawrie. Fault tolerance schemes in shuffle-exchange type interconnection networks. In *Proceedings of 1983 Intl. Conf. On Parallel Processing*, pages 71–75, 1983.

[6] Nicholas Pippenger. Parallel communication with limited buffers. In *Proceedings of FOCS 84*, pages 127–136, 1984.

[7] Stefan Reisch and Georg Schnitger. Three applications of kolmogorov-complexity. In *Proceedings of FOCS 82*, pages 45–52, 1982.

[8] J. T. Schwartz. Ultracomputers. *ACM TOPLAS*, 2:484–521, October 1980.

[9] J. D. Ullman. *Computational aspects of VLSI.* Computer Science Press, 1984.

[10] E. Upfal. Efficient schemes for parallel communication. In *PODC*, pages 55–59, 1982.

[11] L. G. Valiant and G. J. Brebner. Universal schemes for parallel communication. In *Proceedings of STOC 81*, pages 263–277, 1981.

[12] C. Wu and T. Feng. On a class of multistage interconnection networks. *IEEE Transactions on Computers*, C-29:694–702, August 1980.