On the Permutation Capability of Multistage Interconnection Networks

TED H. SZYMANSKI AND V. CARL HAMACHER, SENIOR MEMBER, IEEE

Abstract—**We present analytic models for the blocking probability of both unique path and multiple path multistage interconnection networks under the assumption of either permutation or random memory request patterns. The blocking probability of an interconnection network under the assumption of permutation requests is a quantitative measure of the network's permutation capability. We compare the performance of networks with approximately equivalent hardware complexity. It is shown that variations of banyan networks can be designed with extremely low blocking probabilities under the assumption of permutation requests.**

Index Terms—**Banyan networks, crossbar networks, multistage interconnection networks, performance analysis.**

I. Introduction

As interest in large scale parallel processing systems grows, a number of interconnection network architectures have been proposed. Multistage interconnection networks (MIN's) provide a cost effective alternative to crossbar switches as a means of providing simultaneous (parallel) connections between processors and memory modules. Two fundamental criteria in the selection and design of an interconnection network are parallel connection capability and fault tolerance. In this paper, we study the parallel connection capability of a number of circuit-switched multistage interconnection networks. This capability is considered to characterize the network performance.

Many problems exist in the design of large multiprocessor systems, in particular the design of algorithms to exploit parallelism. Among the effective architectures have been the tightly coupled SIMD and MIMD systems that rely on data skewing algorithms [5], [19], [20], [24], [27]. Data skewing algorithms attempt to distribute the data of a matrix in such a way that when the processors address individual elements of a particular vector (row, column, major diagonal, etc.) in parallel, the resulting memory module request pattern is a permutation of some or all of the module address numbers. That is, in a set of memory requests issued in parallel by the processors, each memory module is addressed by at most one processor. We will refer to this type of request pattern as a *permutation request*. The more general situation, in which a number of processors might request the same memory module in a parallel access request pattern, is defined as a *random request*. When a number of processors simultaneously request access to the same memory module, we assume that only one of these requests can be satisfied, and we say that the other requests are *blocked* at the memory.

Crossbar switches can simultaneously provide all connections required by any permutation request, and are thus said to be *nonblocking*. However, the less expensive MIN's studied in this paper have the property that some individual connections involved in a permutation request may be blocked in the interconnection network. This can happen when a number of inputs to a switch at some stage in the network all request the same output link in moving towards the next stage. Networks that have this property are said to be *blocking* networks. The extent to which blocking occurs in a particular MIN over the set of all permutation requests is clearly an important property.

The *permutation capability* of a MIN generally refers to the fraction of all possible permutation requests that can be realized with no blocking. A related performance measure is the *blocking probability* of a MIN under the assumption that only permutation requests are submitted to it. This blocking probability is defined as the probability that an individual connection request from a processor to a memory module will be blocked, when the set of all possible permutation requests is considered.

Nonblocking switching networks have been the subject of much theoretical research [11], [6]. Of the known networks that are nonblocking for all permutation requests, large crossbars are far too expensive, and the various MIN's require precomputed routing for each individual permutation [6]. Many suboptimal MIN's with simple distributed routing algorithms have been analyzed for their permutation capabilities [24], [19], [18], [3]. An analysis for classes of nonblocking permutation requests that can be realized in the omega network has been presented in [19]. Upper and lower bounds for the number of nonblocking permutation requests that can be realized in the augmented data manipulator (ADM) have been presented in [3], and an analysis of the number of nonblocking permutation requests that can be realized in the generalized shuffle network (GSN) has been presented in [8].

A quantitative measure that has been used for comparing the permutation capability of various interconnection networks is the number of passes a network requires to realize a particular permutation request. Lower time bounds for the simulation of one network by another have been presented in [27]. For GSN's, an analysis of the number of passes required to realize

Manuscript received December 28, 1985; revised July 2, 1986. This work was supported in part by NSERC, Canada, under Grant A-5192.

The authors are with the Departments of Electrical Engineering and Computer Science and the Computer Systems Research Institute, University of Toronto, Toronto, Ont., Canada.

IEEE Log Number 8714104.

a specific permutation request, and an upper bound for the number of passes required to realize arbitrary permutation requests is presented in [4]. A related performance measure is the blocking probability of a network over all permutation requests, as defined above. Franklin has presented simulation results for the blocking probability of banyan networks (built with 2×2 switches) under the assumption of permutation requests [13].

We present analytic models for the blocking probability for various MIN's under permutation requests. The blocking probability over all permutation requests provides a quantitative measure of a network's permutation capability. Section II reviews the definition of banyan networks and some variations of them. Section III analyzes the performance of crossbars and banyan networks of degree k, under random or permutation requests. Section IV analyzes the performance of d-dilated, r-replicated square banyans of degree k, under both request patterns. Section V extends the performance comparison presented by Kruskal and Snir [16] to d-dilated, r-replicated banyans of degree k, also under both request patterns. It is shown that variations of banyan networks can be designed to have extremely low blocking probabilities under permutation requests. Section VI summarizes the paper and contains some concluding remarks.

II. Definitions and a Basic Result

The following definitions have appeared in various forms previously [16]. Assume that the interconnection network connects processors to memories.

A *network* is a directed graph where nodes are of the following three types: 1) *processors* which have indegree 0; 2) *memory modules* (or memories) which have outdegree 0; 3) *switches* which have indegree > 0 and outdegree > 0.

Each (directed) edge represents a unidirectional link going from a node to its successor.

A *banyan network* is defined by Goke and Lipovski [15] to be a network with a unique path from each processor to each memory. This definition implies that the set of paths leading to a node in the network forms a tree and that the set of paths leading from a node also forms a tree. An *n-stage multistage network* is a network in which the switches can be arranged in stages, with all processors attached to the inputs of stage 1 switches, all outputs from stage i are connected to inputs to stage $i + 1$, and all memories are connected to outputs from stage n. Under this definition, we also say that processors are at stage 0, and memories are at stage $n + 1$. Furthermore, we will define the links from processors to stage one switches to be input ports, and the links from stage n switches to memories to be output ports. A *uniform network* is a multistage network in which all switches in the same stage have the same number of input links and the same number of output links. A *square network of degree k* is a network built with $k \times k$ switches (hereafter referred to as a $k^n \times k^n$ banyan). Fig. 1 shows a $2^3 \times 2^3$ banyan network.

The actual routing of a request from a processor (left-side input ports) to a memory module (right-side output ports) in Fig. 1 is easily derived from the binary address of the destination memory module in the usual way. In the Fig. 1

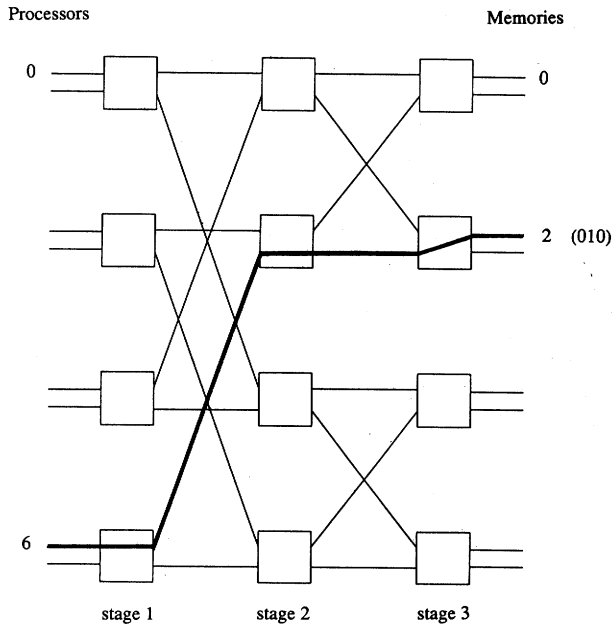


Fig. 1. A $2^3 \times 2^3$ banyan network.

example, 3-bit module addresses are used for routing as follows. The most significant bit selects the upper (0) or lower (1) output of the stage 1 switch to which the request is applied. The next bit selects the stage 2 switch output, and the least significant bit selects the stage 3 switch output port attached to the desired memory module. The example of a request from processor input port 6 to memory module output port 2 (binary = 010) is indicated by thick links in the figure. Generalizations of this type of routing control strategy are assumed throughout the paper.

When two or more requests arrive at one switch and request the same output link, we say that *link contention* has occurred. The contention is *resolved* by selecting one request to be forwarded, with the others being blocked. When this occurs in a stage n switch, we say that *memory conflict* has occurred. In general, the blocking probability of an n-stage network under random patterns consists of a component that reflects the link contentions within the network (called internal links) and a component that reflects the memory conflicts. However, the blocking probability of an n-stage network under permutation request patterns consists of the component that reflects the internal link contentions only, since there are no possible memory conflicts. As observed by Franklin [13], "by addressing ... (the permutation request pattern case) ... the analysis emphasizes the blocking characteristics inherent in the network, excluding blocking effects due to ... (memory) ... conflicts."

While banyan networks are very attractive in terms of their simplicity, considerations such as performance and fault tolerance often dictate the use of more complicated networks. The following two strategies may be used to augment a network G without increasing the complexity of its structure by very much [15], [16]: 1) The *d-dilation* of G is defined to be the network obtained from G by replacing each edge by d distinct edges [see Fig. 2(a)]. A request entering a switch may exit using any of the d edges going to the desired successor

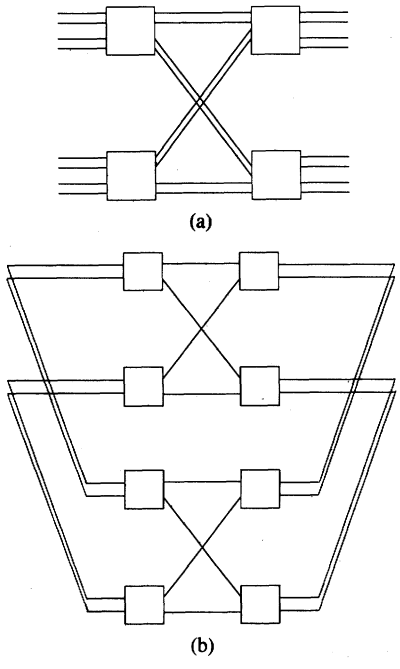


Fig. 2. (a) A 2-dilated $2^2 \times 2^2$ banyan network. (b) A 2-replicated $2^2 \times 2^2$ banyan network.

switch at the next stage. 2) The *r-replication* of G is defined to be the network consisting of r identical copies of G [see Fig. 2(b)]. We do not consider other strategies that can be used to improve the fault tolerance of a network, for example adding extra stages at the inputs to the network [1].

Observe that in each of the two strategies used to improve the fault tolerance of the banyan network, the complexity of the routing algorithms have remained basically unchanged. In addition, the routing algorithms do not depend on the request patterns (i.e., random or permutation) that are submitted to the network, since each request carries its own destination tag which determines the routing decisions made at each switch.

The following lemma regarding the statistical distribution of requests in a network operating under random request patterns has been presented in [16].

Lemma [16]: Let the requests be generated at the processors of a uniform banyan network by independent, identically distributed random processes that uniformly distribute the requests over all memories. Assume that the routing logic at each switch is "fair," i.e., contentions are randomly resolved. Then 1) The patterns of request arrivals at the inputs of the same switch are independent, 2) requests arriving at an input of a switch are uniformly distributed over the outputs of that switch and 3) for each stage in the network, the pattern of request arrivals at the inputs of that stage have the same distribution.

However, under permutation request patterns, the preceding lemma is no longer valid. Our analytic models calculate the probability that an output port of an interconnection network carries a memory request in this case, by addressing the conditional distribution of requests over the outputs of a switch.

Let $p(1, m)$ be the probability that an output link of a node in stage m of an interconnection network carries one request. Hence, each processor generates a request during each cycle with probability $p(1, 0)$. The blocking probability of a network under either random or permutation request patterns will be denoted as pb. Under random patterns, pb will consist of an internal link contention component and a memory conflict component. Under permutation request patterns, pb will consist of an internal link contention component only, since there are no memory conflicts in this case. Note that pb is the probability that an issued request is not satisfied in a particular memory cycle.

Much previous work has focused on system performance in an environment where the following type of data dependencies exist: when a processor issues a memory request that is not satisfied in one cycle, it will resubmit the same request during the next cycle (the *data dependency* assumption) [9]. Previous studies [25], [23], [9], [7] have indicated that the steady-state blocking probability under random request patterns does not change much when we drop the data dependency assumption and assume that requests are random and independent during each cycle (i.e., unsatisfied requests are ignored). This assumption that unsatisfied requests are ignored, commonly called the *regenerative* assumption, leads to a simple closed form expression for the blocking probability of a crossbar switch under random request patterns.

However, the regenerative assumption is not valid under permutation request patterns; systems using data skewing algorithms currently operate with this data dependency, hence unsatisfied requests must be resubmitted during the next cycle. In an MIMD system, the resubmitted requests may conflict with the newly issued requests and hence the steady-state request patterns will not be permutations. However, it would appear that the performance of these systems will be maximized when the blocking probability of the network under permutation requests is minimized.

In an SIMD system, all processors must wait until each processor receives the data it has requested; a number of memory cycles may be required to satisfy all the requests. It would also appear that the performance of these systems will be maximized when the blocking probability of the network under permutation requests is minimized. Note that the data skewing algorithms and the organization of the system will determine the type of permutation patterns generated in an SIMD system. Lawrie has presented an organization in which the permutation patterns generated by the processors are nonblocking in the omega network [19].

We do not address data dependencies in this paper, and simply assume that unsatisfied requests are ignored. Our analytic models give a quantitative measure of the parallel connection capability of a network under the assumption of permutation request patterns, and can be used as an approximate figure of merit for a network's suitability in applications where the permutation request pattern is dominant.

III. Analysis of Crossbar and Banyan Networks

Patel has presented analytic models for circuit-switched crossbars and banyan networks [23]. We present an alternate derivation here using our notation. The assumptions for the analysis are as follows: 1) during the beginning of each cycle

every processor issues a request with probability $p(1, 0)$; 2) requests are randomly and uniformly distributed among the memories; and 3) unsatisfied requests in any cycle are ignored, and a new set of requests is issued during the next cycle subject to 1) and 2). As mentioned earlier, the last assumption is made to simplify the analysis: however, analysis and simulations [23] indicate that the performance is only slightly different when the data dependency assumption is retained.

Assume a crossbar switch of size $k \times k$, i.e., a 1-stage network consisting of a single switch and k processors and k memories. The requests arriving at the switch during a particular cycle are statistically independent. In a given cycle, the probability that a particular input port receives a request to a particular output port (memory) is then $p(1, 0)/k$. The probability that a particular input port does not receive a request for a particular output port is then $1 - p(1, 0)/k$. Since input events are independent, the probability that a particular output port is not requested by any processor is then

$$\left(1-\frac{p(1, 0)}{k}\right)^k.$$

Hence, the probability that an output port is selected by at least one processor is

$$p(1, 1)=1-\left(1-\frac{p(1, 0)}{k}\right)^k. \tag{1}$$

Finally, the fraction of actual processor requests that are blocked, which is synonymous with the blocking probability, is given by

$$pb=\frac{p(1, 0)-p(1, 1)}{p(1, 0)}$$
$$=1-p(1, 1)/p(1, 0).$$

Using the same arguments as Patel [23], the blocking probability in $k^n \times k^n$ banyans under random request patterns can also be developed. The individual switching nodes are $k \times k$ crossbars. Observing that the outputs of crossbar switches in stage m become inputs to crossbar switches in stage $m + 1$, the recurrence relation

$$p(1, m+1)=1-\left(1-\frac{p(1, m)}{k}\right)^k \tag{2}$$

can be used to generate the blocking probability

$$pb=1-p(1, n)/p(1, 0).$$

Since delta networks [23], omega networks [19], indirect binary cube networks [24] are all banyan networks, this analysis applies to these networks as well.

A. Alternative Analysis for Crossbars and Banyans

The previous analytic models are often used to calculate quantitative measures of a network's performance, and are considerably simpler than the models that preceded them [7], [9], [25]. However, "real programs are not random in their addressing patterns" [25]. In order to account for particular memory request distributions, we need different analytic models. We now present an alternative analysis for crossbars and banyans under the same assumptions as the previous section, i.e., the requests are randomly distributed. The resulting analytic models, while slightly more complicated than the above, are in a form that can then be adapted for permutation requests. The adaptation will be done in the next section.

Assume that the events at the inputs of a $k \times k$ crossbar switch are independent. For $p(1, 0)$ as before, the probability that i requests arrive on k input ports is given by

$$\binom{k}{i} p(1, 0)^i(1-p(1, 0))^{k-i}. \tag{3.1}$$

We now calculate an expression for $p(1, 1)$, the probability that an output port is selected by at least one processor. Suppose that i requests arrive at the crossbar; we want an expression for the probability that j of these requests select a particular output port. Given that i requests arrive, the number of ways of selecting j of these is simply i *choose* j. Since the requests are random and independent, these j requests select one of k output ports with probability $(1/k)^j$. The remaining $i - j$ requests select other output ports with probability $(1 - (1/k))^{i-j}$. Hence, the probability that one or more requests select the same output port, given that i requests arrive at the inputs to the crossbar, is given by

$$\sum_{j=1}^{i}\binom{i}{j}\frac{1}{k^j}\left(1-\frac{1}{k}\right)^{i-j}. \tag{3.2}$$

Hence, the blocking probability of a $k \times k$ crossbar is given by

$$pb=1-p(1, 1)/p(1, 0)$$

where

$$p(1, 1)=\sum_{i=1}^{k}\binom{k}{i} p(1, 0)^i(1-p(1, 0))^{k-i}$$
$$\cdot \sum_{j=1}^{i}\binom{i}{j}\frac{1}{k^j}\left(1-\frac{1}{k}\right)^{i-j}. \tag{3.3}$$

Using the same arguments as earlier, the recurrence relation

$$p(1, m+1)=\sum_{i=1}^{k}\binom{k}{i} p(1, m)^i(1-p(1, m))^{k-i}$$
$$\cdot \sum_{j=1}^{i}\binom{i}{j}\frac{1}{k^j}\left(1-\frac{1}{k}\right)^{i-j} \tag{4}$$

leads to the blocking probability

$$pb=1-p(1, n)/p(1, 0)$$

for $k^n \times k^n$ banyans.

The above analytic model has an explicit term that accounts

for the probability that one or more requests select a particular output port (3.2). This explicit term can be modified to account for permutation request patterns.

B. Analysis of Crossbars and Banyans under Permutation Requests

In this section, we present an approximate analysis of the blocking probability of $k^n \times k^n$ banyan networks under the assumption of permutation request patterns. Recall that this blocking probability will reflect the blocking inherent in the network due to internal link contention only, since there are no memory conflicts by assumption.

The assumptions for the analysis are as follows: 1) during the beginning of each cycle every processor issues a request with probability $p(1, 0)$; 2) in the set of memory modules requested by the processors, each module is requested at most once (a permutation request pattern); and 3) unsatisfied requests in any cycle are ignored, and a new set of requests is issued during the next cycle subject to 1) and 2). We begin by applying the general technique to the degenerate case of a crossbar ($n = 1$), where we know the blocking probability is 0.

1) Crossbars: Assuming that the events at the inputs of a crossbar switch are independent, then the probability that i requests arrive on the k input ports is given by (3.1).

We calculate an expression for $p(1, 1)$, the probability that an output port is selected by at least one processor. Suppose that i requests arrive at the crossbar; we want an expression for the probability that j of these requests select a particular output port. When the requests are random and independent, this probability was given by (3.2). When the requests are permutations, (3.2) must be modified slightly, as follows. Given that i requests arrive, the number of ways of selecting j of these is simply i *choose* j. Since the request pattern is a permutation request, only one request can select a particular port and hence $j = 1$ is the only nonzero case to consider. This one request selects a particular output port with probability $1/k$. Given that i requests arrive, and $j = 1$ requests select a particular output port, the remaining $i - j$ requests select other output ports with probability one. Hence, the probability that one or more requests select the same output port, given that i requests arrive at the inputs to the crossbar, is given by i *choose* $1 \cdot (1/k)$. Hence, (3.3) can easily be modified to yield the probability that an output port of a $k \times k$ crossbar receives a request, under the permutation request assumption. The resulting model is given by

$$p(1, 1) = \sum_{i=1}^{k} \binom{k}{i} p(1, 0)^i (1 - p(1, 0))^{k-i} \binom{i}{1} \frac{1}{k} = p(1, 0)$$

$$pb = 1 - p(1, 1)/p(1, 0) = 0.$$

This result is rather obvious, but it illustrates the technique of biasing probabilities that will be used in the analysis of banyan networks.

2) Banyan Networks: We now present an approximate analysis of $k^n \times k^n$ banyan networks under permutation requests. Equation (4) will be modified to account for the probability that two or more requests select the same internal output link of a $k \times k$ crossbar switch in stage m. First, it is illustrative to consider an analogy with a shuffled deck of cards.

Consider the probabilities of selecting aces from a shuffled deck of cards without replacing the selected cards. The first card selected will be an ace with a probability of 4/52. The second card selected will be an ace with probability 3/51, given that the first card selected was an ace. The second card selected will be an ace with probability 4/51, given that the first card selected was not an ace. Let n_{ace} be the number of aces already selected. Let $n_{\text{not\_ace}}$ be the number of cards already selected that are not aces. Then the probability that the next card selected will be an ace is given by

$$\frac{4 - n_{\text{ace}}}{52 - n_{\text{ace}} - n_{\text{not\_ace}}}.$$

Now consider the probabilities of requests selecting outputs at a particular $k \times k$ crossbar switch in stage m of the banyan network. Each output link leads to k^{n-m} memory modules. The first request selects a particular output link with probability k^{n-m}/k^{n-m+1}. Given that the first request selects a particular output link, then the second request selects that same output link with probability $(k^{n-m} - 1)/(k^{n-m+1} - 1)$.

As a notational convenience, define function $p\_s(m, s, r)$ as the probability that a request at a switch in stage m ($m < n$) will select a particular output link given that s requests have already selected that link, and that r requests have already selected other links.

$$p\_s(m, s, r) = \begin{cases} \dfrac{k^{n-m} - s}{k^{n-m+1} - s - r}, & \text{for } s < k^{n-m},\ s + r < k^{n-m+1} \\ 0, & \text{otherwise.} \end{cases}$$

In a similar manner, define function $p\_ns(m, s, r)$ as the probability that a request at a switch in stage m ($m < n$) will not select a particular output link given that s requests have already selected that link, and that r requests have already selected other links.

$$p\_ns(m, s, r) = \begin{cases} \dfrac{(k-1)k^{n-m} - r}{k^{n-m+1} - s - r}, & \\ & \text{for } r < (k-1)k^{n-m},\ s + r < k^{n-m+1} \\ 0, & \text{otherwise.} \end{cases}$$

Assuming that the events occurring at the inputs of each switch in stage m of the banyan network are independent (which will result in a slightly pessimistic blocking probability), then the probability that i requests arrive on k input links of a switch in stage m is given by

$$\binom{k}{i} p(1, m)^i (1 - p(1, m))^{k-i}.$$

The probability that j requests select a particular output is given by

$$\prod_{s=0}^{j-1} p\_s(m, s, 0).$$

The probability that $i - j$ requests do not select that particular output link, given that j requests have already selected it, is given by

$$\prod_{r=0}^{i-j-1} p\_ns(m, j, r).$$

Hence, (4) can easily be modified to yield the blocking probability of $k^n \times k^n$ banyan networks under permutation requests. The resulting recurrence relation is given by

$$p(1, m+1) = \sum_{i=1}^{k} \binom{k}{i} p(1, m)^i (1 - p(1, m))^{k-i}$$

$$\cdot \sum_{j=1}^{i} \binom{i}{j} \prod_{s=0}^{j-1} p\_s(m, s, 0) \prod_{r=0}^{i-j-1} p\_ns(m, j, r) \quad (5)$$

$$pb = 1 - p(1, n)/p(1, 0).$$

This approximation, labeled approximation (5), is plotted against simulation results for $2^n \times 2^n$ and $4^n \times 4^n$ banyans in Fig. 3. Our simulation results have a 95 percent confidence interval of one half of one percent of the simulated value, and they agree very closely with Franklin's simulations for the $2^n \times 2^n$ case [13].

Our approximation for the blocking probability is slightly pessimistic; for $2^n \times 2^n$ banyans the approximation is consistently larger than the simulation results, and consistently exceeds the 95 percent confidence intervals; the maximum error is 2.4 percent and the average error is 1.6 percent (of the simulation values). However, for $4^n \times 4^n$ banyans the approximation is much closer, although still slightly pessimistic; the maximum and average errors are 0.4 percent and 0.31 percent, respectively.

The discrepancies are likely due to the assumption of statistical independence of the events occurring at the input links of a switch in each stage. To test this hypothesis, we present a refined analytic model that accounts for the *first-order* effects of statistical dependence in the next section.

3) A Refined Analysis of Banyan Networks Under Permutation Requests: A refined approximation that accounts for the *first-order* effects of statistical dependence is as follows. The previous analysis calculates $p(1, m)$, the probability that the output link of a switch in stage m carries a request, and assumes that this result is valid for all switches in stage m. The probability that i requests arrive at the inputs of a switch in stage $m + 1$ was calculated using the binomial distribution, which assumes that the events occurring on each input are independent.

A refined analysis must calculate the probability that i requests arrive at the inputs of a switch in stage $m + 1$ based on the events occurring in the previous stage m. To simplify the presentation, we consider only $2^n \times 2^n$ banyans; the results are easily generalizable to $k^n \times k^n$ banyans.

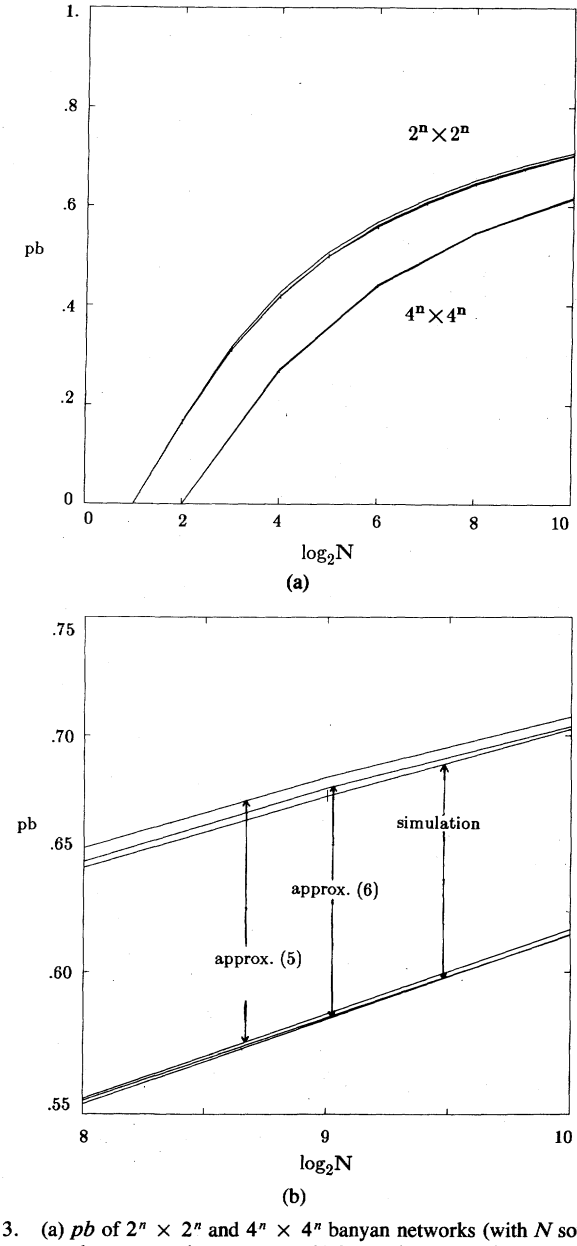


Fig. 3. (a) pb of $2^n \times 2^n$ and $4^n \times 4^n$ banyan networks (with N sources) under permutation requests. (b) Expanded view of (a).

Let $p(i, m)$, $i = 0, 1, 2$, be the probabilities that i requests arrive at the two inputs to a switch in stage m. Assume that each 2×2 switch at the input of the network receives two requests with probability one, i.e., $p(2, 1) = 1$, and $p(0, 1) = p(1, 1) = 0$. Given $p(i, m)$, we can calculate $p(i, m + 1)$ accounting for the first-order effects of statistical dependence. Recalling that the last stage of the network will not block, given $p(i, n)$, where n is the last stage of the network, we can calculate the blocking probability. The probability that an output port will be selected by one processor is given by

$$\frac{1}{2} \sum_{i=1}^{2} p(i, n) \cdot i$$

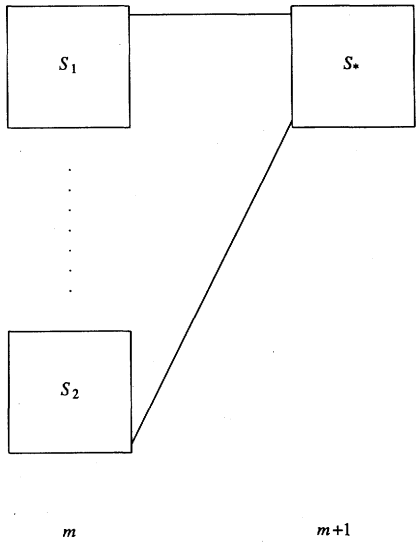


Fig. 4. Stage $m + 1$ of a $k^n \times k^n$ banyan network.

and hence the blocking probability is simply

$$pb = 1 - \frac{1}{2}\sum_{i=1}^{2} p(i, n) \cdot i.$$

We now must calculate $p(i, m + 1)$ given $p(i, m)$. Consider a switch s^* in stage $m + 1$ of a $2^n \times 2^n$ banyan network, as shown in Fig. 4. Switch s^* can be reached from 2 switches in stage m (by traversing an output link). Call these two switches s_1 and s_2.

Each switch s_1 or s_2 can have 0, 1, or 2 requests arriving at its inputs, and these events occur with probability $p(0, m)$, $p(1, m)$, and $p(2, m)$, respectively. Hence, there are $3 \times 3 = 9$ possible states to consider. The probability that i requests arrive at s_1 and j requests arrive at s_2 is simply $p(i, m) \cdot p(j, m)$.

Consider the case where s^* has no requests arriving at its inputs. (The cases where one or two requests arrive can be handled in a similar manner.) All requests in switches s_1 and s_2 must not select a link leading to s^*.

If s_1 has i requests arriving at its inputs, these i requests do not select the link leading to s^* with probability

$$\prod_{h=1}^{i} p\_ns(m, 0, h-1).$$

If s_2 has j requests at its inputs, these j requests do not select s^* with probability

$$\prod_{h=1}^{j} p\_ns(m, 0, i+h-1).$$

Note that the probability that requests in switch s_2 do not select switch s^* depends on the events occurring in switch s_1 (in particular, the fact that i requests from switch s_1 have also not selected s^*), thus accounting for the first-order effects of statistical dependence.

Hence, the probability that s^* has no requests arriving at its inputs is

$$p(0, m+1) = \sum_{i=0}^{2}\sum_{j=0}^{2} p(i, m) \cdot p(j, m) \cdot \prod_{h=1}^{i} p\_ns(m, 0, h-1) \prod_{h=1}^{j} p\_ns(m, 0, i+h-1). \quad (6)$$

The cases where switch s^* has 1 and 2 requests arriving at its inputs can be derived in a similar manner. The recurrence relations for these cases are

$$p(1, m+1) = 2 \cdot \sum_{i=1}^{2}\sum_{j=0}^{2}\sum_{s=1}^{i} p(i, m) \cdot p(j, m) \binom{i}{s} \cdot \prod_{x=1}^{s} p\_s(m, x-1, 0) \cdot \prod_{y=1}^{i-s} p\_ns(m, s, y-1) \cdot \prod_{z=1}^{j} p\_ns(m, s, z-1+i-s)$$

$$p(2, m+1) = \sum_{i=1}^{2}\sum_{j=1}^{2} p(i, m) \cdot p(j, m) \cdot \sum_{s=1}^{i} \left(\binom{i}{s} \prod_{x=1}^{s} p\_s(m, x-1, 0) \cdot \prod_{y=1}^{i-s} p\_ns(m, s, y-1) \cdot \sum_{z=1}^{j} \left(\binom{j}{z} \prod_{x=1}^{z} p\_s(m, z-1+s, i-s) \cdot \prod_{y=1}^{j-z} p\_ns(m, s+z, y-1+i-s)\right)\right).$$

The analysis can also be generalized to $k \times k$ switches. (Since the number of cases to consider grows rapidly with the switch size, a generalized recursive procedure can be used.)

The results of the refined approximation, labeled approximation (6), are plotted against simulations of $2^n \times 2^n$ and $4^n \times 4^n$ banyans in Fig. 3. For the $2^n \times 2^n$ banyan, the maximum and average errors are 0.6 percent and 0.36 percent, respectively, compared to 2.4 and 1.6 percent for approximation (5). For the $4^n \times 4^n$ banyan, the maximum and average errors are 0.25 and 0.11 percent, respectively, compared to 0.4 and 0.31 percent for approximation (5).

Hence, it appears that the cause of the discrepancies between approximation (5) and the simulation results is the assumption that the events occurring on the inputs of a switch are statistically independent. However, approximation (5) is simpler and computationally more efficient than the refined analysis and offers nearly identical results.

IV. Performance Analysis for d-Dilated, r-Replicated $k^n \times k^n$ Banyan Networks

Kruskal and Snir [16] have analyzed d-dilated $2^n \times 2^n$ banyan networks and r-replicated $k^n \times k^n$ banyan networks under random requests. We present a summary of their results in the next two sections. We then generalize these models for arbitrary d-dilated, r-replicated $k^n \times k^n$ banyan networks, under random requests, and then extend them to yield approximations for these networks under permutation requests.

A. Performance Analysis for d-Dilated $2^n \times 2^n$ Banyan Networks

Kruskal and Snir [16] have analyzed d-dilated $2^n \times 2^n$ banyan networks under the following assumptions: 1) during the beginning of each cycle every processor issues a request with probability one; 2) requests are randomly and uniformly distributed over the memories; 3) unsatisfied requests during any cycle are ignored, and a new set of requests is issued during the next cycle subject to 1) and 2); and 4) if $m > d$ requests arrive at a switch and compete for d (output) edges, then d of them are selected randomly and forwarded, and the remainder are ignored (i.e., blocked).

Let $p(i, m)$ be the probability that i requests occupy d identical edges leaving a switch in stage m. The probability that i requests arrive at a switch in stage $m+1$ is $\Sigma_{r+s=i} p(r, m) \cdot p(s, m)$. The probability that j of these requests are directed to a particular output link is (i choose j)2^{-i}. The initial and recurrence relations are

$$p(j, 0) = 0, \qquad \text{for } j \neq 1$$

$$p(1, 0) = 1.$$

For $j < d$

$$p(j, m+1) = \sum_{i=j}^{2d} \left(\sum_{r+s=i} p(r, m) \cdot p(s, m) \right) \binom{i}{j} 2^{-i}, \quad (7)$$

and for $j = d$

$$p(j, m+1) = \sum_{i=d}^{2d} \left(\sum_{r+s=i} p(r, m) \cdot p(s, m) \right) \sum_{t=d}^{i} \binom{i}{t} 2^{-i}.$$

B. Performance Analysis for r-Replicated $k^n \times k^n$ Banyan Networks

Kruskal and Snir [16] have also analyzed r-replicated $k^n \times k^n$ banyan networks under the same assumptions as the previous section. An r-replicated $k^n \times k^n$ banyan consists of r copies of $k^n \times k^n$ banyan networks, and (2) applies to each copy. Assume that every processor issues a request during each cycle with probability $p'(1, 0)$, and randomly sends it to one of the r copies. The events occurring at the inputs to each copy are statistically dependent; if a request is sent to one particular copy, then it will not be sent to any other copy. However, assuming that the r copies are independent, which is slightly optimistic, then the probability that a request is issued at the input to one particular plane is $p'(1, 0)/r$, and the probability that the final stage of the plane has a request is given by (2). The blocking probability of the entire network is then

$$p(1, 0) = p'(1, 0)/r$$

$$p(1, m+1) = 1 - \left(1 - \frac{p(1, m)}{k}\right)^k$$

$$p(0, n) = 1 - p(1, n)$$

$$pb = 1 - \frac{1 - p(0, n)^r}{p'(1, 0)} \quad (8)$$

The performance of r-replicated banyans can be improved slightly by changing the assumption that issued requests are randomly sent to one of the r copies. Note that if k divides r, an r-replicated banyan can be configured so that the first $\log_k r$ stages do not block. The analysis of each copy is equivalent to assuming each copy has $\log_k N - \log_k r$ stages. The r copies are now independent, so the analysis is exact.

C. Performance Analysis for d-Dilated $k^n \times k^n$ Banyan Networks

The preceding model (7) for d-dilated $2^n \times 2^n$ banyan networks can easily be generalized to account for d-dilated $k^n \times k^n$ banyans, with the extra assumption: 1) during the beginning of each cycle, every processor issues a request with probability $p'(1, 0)$.

The events occurring on each of the k logical input links of a logical $k \times k$ switch must be examined. Define an array of variables r_p, $p \in 1 \cdots k$, where r_p equals the number of requests that arrive on the pth logical input link of a logical $k \times k$ switch in stage m. For example, if each of k logical input links carries zero requests, then each $r_p = 0$, and this event occurs with probability $p(0, m)^k$. The probability that i requests arrive at a switch in stage m is

$$\left(\sum_{\Sigma_x r_x = i} \prod_x p(r_x, m) \right).$$

The probability that j of these requests are directed to a particular output link is

$$\binom{i}{j} \frac{1}{k^j} \left(1 - \frac{1}{k}\right)^{i-j}.$$

Hence, the initial and recurrence relations for a d-dilated $k^n \times k^n$ banyan are

$$p(1, 0) = p'(1, 0)$$

$$p(0, 0) = 1 - p(1, 0)$$

$$p(j, 0) = 0, \qquad \text{for } j \neq 1 \text{ and } j \neq 0.$$

For $j < d$

$$p(j, m+1) = \sum_{i=j}^{kd} \left(\sum_{\Sigma_x r_x = i} \prod_x p(r_x, m) \right) \binom{i}{j} \frac{1}{k^j} \left(1 - \frac{1}{k}\right)^{i-j} \quad (9)$$

and for $j = d$

$$p(j, m+1)=\sum_{i=j}^{kd}\left(\sum_{\Sigma_x r_x=i}\prod_x p(r_x, m)\right)\cdot\sum_{t=d}^{i}\binom{i}{t}\frac{1}{k^t}\left(1-\frac{1}{k}\right)^{i-t}.$$

D. Performance Analysis for d-Dilated, r-Replicated $k^n \times k^n$ Banyan Networks

While (8) accounts for r-replicated $k^n \times k^n$ banyan networks, it is desirable to integrate r-replications into (9). Equation (9) can easily be modified to account for d-dilated, r-replicated $k^n \times k^n$ banyans, under either of the earlier assumptions regarding r-replications (i.e., whether the first few stages can block or not). Assuming that the first $\lfloor \log_k r \rfloor$ stages of an r-replication do not block, then an exact analysis for d-dilated, r-replicated $k^n \times k^n$ banyans is derived by changing the initial and termination conditions of (9):

$$p(1, 0)=p'(1, 0)/r$$

$$p(0, 0)=1-p(1, 0)$$

$$p(j, 0)=0, \qquad \text{for } j\neq 0 \text{ and } j\neq 1$$

$$pb=1-\frac{1-p(0, n-\lfloor \log_k r \rfloor^r)}{p'(1, 0)}. \tag{10}$$

1) Performance Analysis for d-Dilated, r-Replicated $k^n \times k^n$ Banyan Networks Under Permutation Requests: We now derive approximations for d-dilated, r-replicated $k^n \times k^n$ banyan networks under permutation requests by modifying (7) in a manner analogous to that used in Section III-B-2.

Assume that the events occurring at the inputs of a switch are statistically independent, which will result in a slightly pessimistic blocking probability. The probability that i requests arrive at a switch in stage m is

$$\sum_{\Sigma_x r_x=i}\prod_x p(r_x, m).$$

The probability that j of these requests are directed to a particular output link and $i - j$ requests are not directed to the same output link is

$$\binom{i}{j}\prod_{x=0}^{j-1} p\_s(m, x, 0)\prod_{y=0}^{i-j-1} p\_ns(m, j, y)$$

where functions p_s and p_ns were defined in Section III-B-2.

Under permutation requests, the last stage of a network will not block any requests. In general, under permutation requests, the last ls stages of a network will not block any requests, where ls is given by

$$ls=\lfloor \log_k d \rfloor+1.$$

To minimize the errors introduced by the statistical independence assumption, we terminate the iteration when the remaining stages are known not to block. Hence, the initial and recurrence relations are given by

$$p(1, 0)=p'(1, 0)/r$$

$$p(0, 0)=1-p(1, 0)$$

$$p(j, 0)=0, \qquad \text{for } j\neq 0 \text{ and } j\neq 1.$$

For $j \neq 0$ and $j < d$

$$p(j, m+1)=\sum_{i=j}^{kd}\left(\sum_{\Sigma_x r_x=i}\prod_x p(r_x, m)\right)\binom{i}{j}\cdot\prod_{x=0}^{j-1} p\_s(m, x, 0)\prod_{y=0}^{i-j-1} p\_ns(m, j, y). \tag{11}$$

For $j = d$

$$p(j, m+1)=\sum_{i=j}^{kd}\left(\sum_{\Sigma_x r_x=1}\prod_x p(r_x, m)\right)\sum_{t=d}^{i}\binom{i}{t}\cdot\prod_{x=0}^{t-1} p\_s(m, x, 0)\prod_{y=0}^{i-t-1} p\_ns(m, j, y).$$

For $j = 0$

$$p(j, m+1)=1-\sum_{i=1}^{d} p(i, m+1)$$

For $n >= ls$

$$p(1, n)=\frac{1}{d}\sum_{i=1}^{d} p(i, n-ls)-i$$

$$p(0, n)=1-p(1, n)$$

$$pb=1-\frac{p(1, n)}{p(1, 0)}.$$

This approximation is plotted against a number of simulations in the next section. Note that this approximation is the most general analytic model we present, and it encompasses all previously presented models for the blocking probability of multistage networks under permutation requests (with the exception of the refined analysis in Section III-B-3 that accounted for the statistical dependence).

V. Performance and Permutation Capability Comparisons

We first compare the blocking probability of unique path banyan networks under both random and permutation requests. In all the following graphs, assume that each processor always issues a request during each cycle. Any simulation curves will be shown with confidence interval bars. Curves without these bars are analytic results. However, many simulations are indistinguishable from the analytic results on these graphs.

Fig. 5(a) illustrates the blocking probability of a crossbar and of unique path banyan networks under random requests. Fig. 5(b) illustrates the blocking probability of unique path banyan networks under permutation requests. As we may intuitively expect, banyans that perform better under random requests also perform better under permutation requests; the

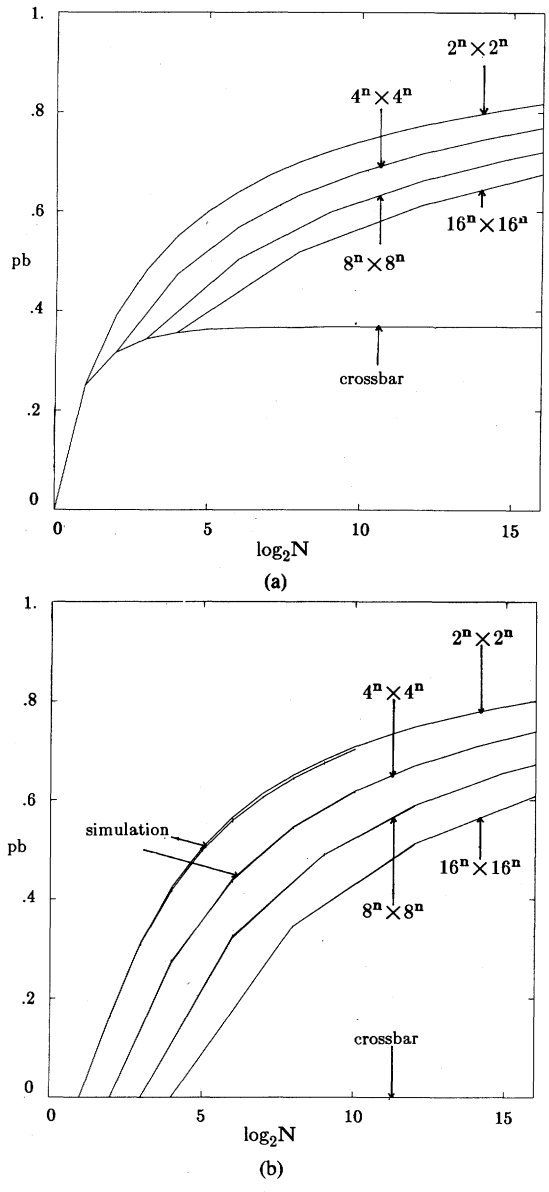


Fig. 5. (a) *pb* of $k^n \times k^n$ banyans (with N sources) under random requests. (b) *pb* of $k^n \times k^n$ banyans (with N sources) under permutation requests.

relative positions of the blocking probability curves remain approximately unchanged when the request patterns change from random to permutation. It appears that as the network size increases the blocking probability under permutation requests approaches that under random requests. It has been shown that the blocking probability of unique path banyan networks under random requests asymptotically approaches one [16]. It appears that the same result applies under permutation requests.

We now examine the effects that dilations and replications have on the blocking probability, under permutation requests. Fig. 6(a) compares *pb* versus $\log_2 N$ for varying dilations of $2^n \times 2^n$ banyans. Fig. 6(b) compares *pb* versus $\log_2 N$ for varying dilations of $4^n \times 4^n$ banyans. Fig. 6(c) compares *pb* versus $\log_2 N$ for varying replications of $2^n \times 2^n$ banyans. Fig. 6(d) compares *pb* versus $\log_2 N$ for varying replications of $4^n \times 4^n$ banyans. (Some of these switches may be currently unrealizable due to pin limitations.)

It appears that for switches of degree 2 or 4, a dilation of 4 is more than adequate to ensure a very low blocking probability under permutation requests, and any further dilations have little effect.

Performance comparisons of d-dilated $2^n \times 2^n$ banyans and r-replicated $k^n \times k^n$ banyans have been presented by Kruskal and Snir in [16]. They conclude that for practical numbers of processors both techniques have comparable performances and other factors, such as delay, fault tolerance, and layout would likely be the determining factors. We extend the performance comparison of [16] to d-dilated, r-replicated $k^n \times k^n$ banyans. In addition, the permutation capabilities of these networks are compared.

It is possible to build circuit-switched banyan networks with different performances by increasing the number of ports on each switch while proportionally decreasing the bandwidth of each port [16], [14]. The same does not necessarily apply for packet-switched banyans, which may require at least two control lines (request/acknowledge) per port [14]. By varying the number of ports per switch, the bandwidth per port and the depth of the network may be adjusted. The depth of a network will determine its cycle time. These parameters, in addition to the expected blocking probability, will determine the true network performance. As in [16], our comparisons are based on blocking probability and permutation capability only, for networks built with switches of comparable port complexity.

Fig. 7(a)–(c) compares the blocking probability and permutation capability of 2-dilated, 4-dilated, and 8-dilated $2^n \times 2^n$ networks with r-replicated networks of comparable hardware complexity. Also, in each graph only switches with the same number of input/output ports are used.

Fig. 7(a) compares the blocking probability and permutation capability for 2-dilated $2^n \times 2^n$ and 4-replicated $4^n \times 4^n$ banyans [16]. As in [16], our model of r-replication assumes that a processor randomly selects which of the r copies the request will be issued to.

Fig. 7(b) compares the 4-dilated $2^n \times 2^n$ and 12-replicated $8^n \times 8^n$ banyans from [16]. These switches have the same pin requirements and hence the same bandwidth.

Fig. 7(c) compares the 8-dilated $2^n \times 2^n$ and 32-replicated $16^n \times 16^n$ banyans in [16] with a 4-dilated $4^n \times 4^n$ banyan. The latter network has approximately one quarter of the hardware cost of the first two networks.

A number of observations are in order. It appears that networks that perform comparable to a crossbar under random requests may or may not perform comparable to a crossbar under permutation requests. Fig. 7(b) and (c) illustrates a number of networks that exhibit performance comparable to a crossbar for practical N (i.e., $N \leq 64K$) under random requests and that vary widely in their performances under permutation requests. Recall that under random requests, the blocking probability consists of a component reflecting link contention within the network (the *inherent network blockage*), and a component reflecting memory conflicts at the outputs of the network. Under random requests, it appears that if the inherent network blockage is less than the asymptotic blockage $1/e$ of a crossbar (which is strictly due to memory conflicts) then it is "masked" by the blockage component due to memory conflicts. When the blockage component due to memory conflicts is removed with the permutation assump-

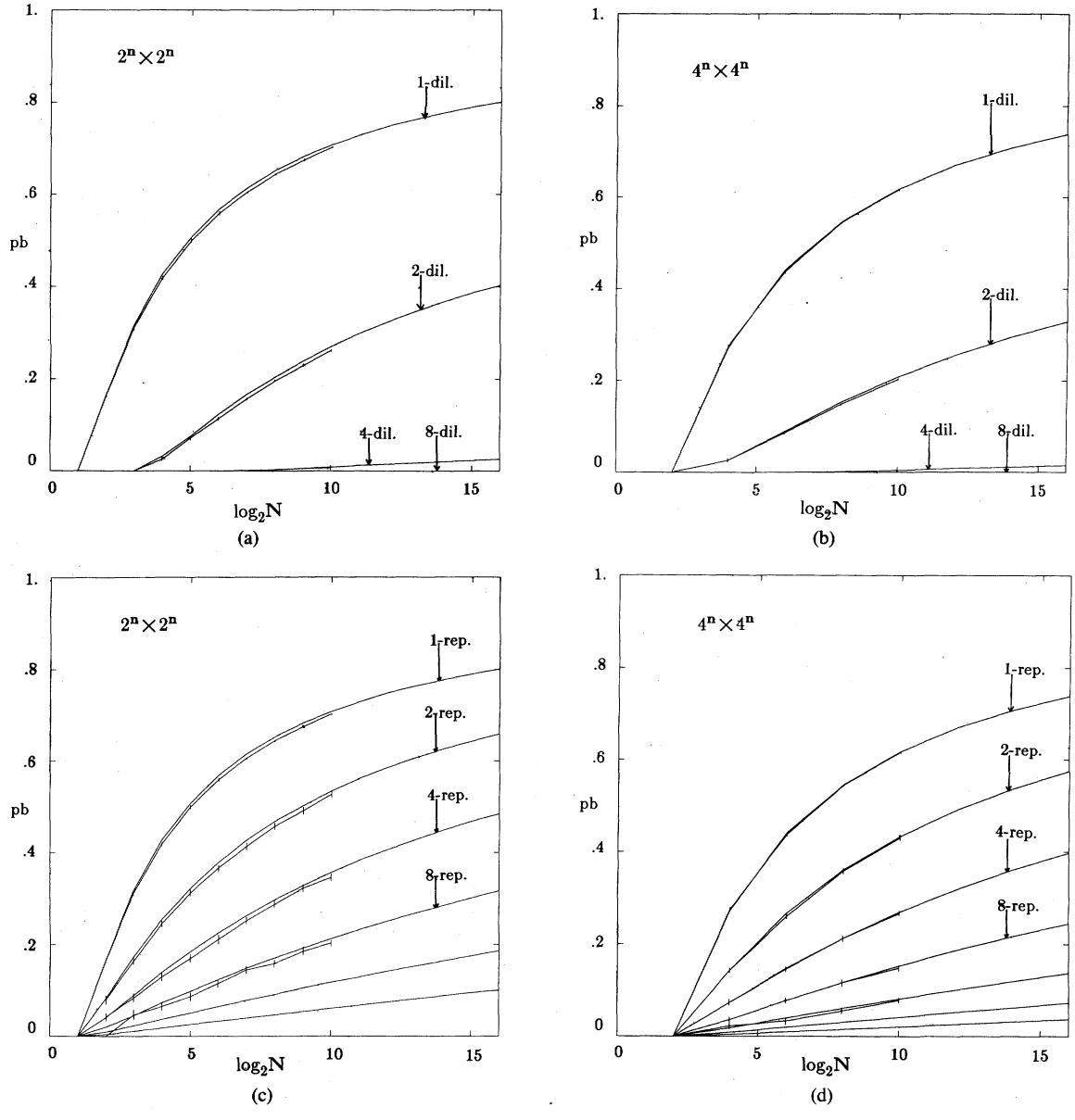


Fig. 6. (a) *pb* for various dilations of $2^n \times 2^n$ banyans (with N sources). (b) *pb* for various dilations of $4^n \times 4^n$ banyans (with N sources). (c) *pb* for various replications of $2^n \times 2^n$ banyans (with N sources). (d) *pb* for various replications of $4^n \times 4^n$ banyans (with N sources).

tion, the inherent network blockage becomes apparent and may vary significantly.

Second, the observation in [16] that dilated and replicated networks have similar performances, and that other factors may determine a network's suitability for a particular situation, is not totally correct. In Fig. 7(c), the 4-dilated $4^n \times 4^n$ banyan offers comparable performance under random or permutation requests compared to the others, at approximately one quarter of the switch cost and less than one quarter of the link cost. It would appear that dilated networks are far more cost effective than replicated networks. (We have ignored the cost of multiplexors at the inputs and outputs of all networks.)

Finally, d-dilated, r-replicated banyan networks can be configured to offer extremely low blocking probabilities under permutation requests, making them suitable for supercomputer architectures, in particular SIMD/MIMD supercomputers using data skewing algorithms. In Fig. 7(c), the 4-dilated $4^n \times 4^n$ is not only far more cost effective than the other configurations, it offers blocking probabilities of 0.0048 and 0.016 at $N = 1024$ and $N = 65536$, respectively, under permutation requests. The larger dilations offer even lower blocking probabilities; the 8 dilation of a $4^n \times 4^n$ banyan has a blocking probability very nearly zero. Fig. 8 illustrates an expanded view of the blocking probability for 4-dilated $2^n \times 2^n$ and 4-dilated $4^n \times 4^n$ banyans under permutation requests.

In architectures using data-skewing algorithms, lowering *pb* decreases the number of requests that need to be resubmitted in the next cycle. In MIMD systems, the steady-state request patterns will not be "perfect permutations" since the resubmitted requests may conflict with the newly issued requests. However, by keeping the blocking probability under permutation requests low, the number of resubmissions will be

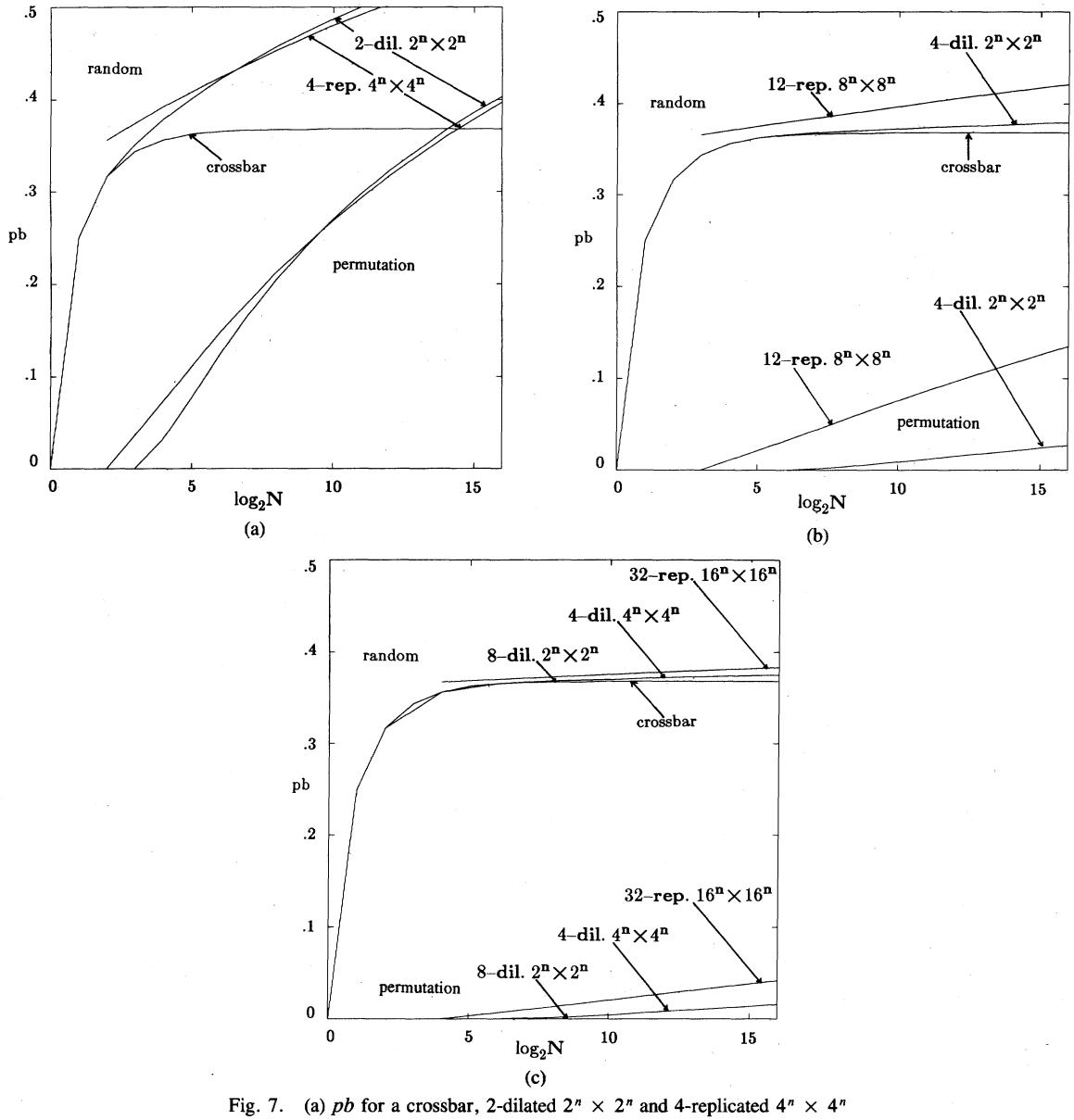


Fig. 7. (a) *pb* for a crossbar, 2-dilated $2^n \times 2^n$ and 4-replicated $4^n \times 4^n$ banyans (with N sources). (b) *pb* for a crossbar, 4-dilated $2^n \times 2^n$ and 12-replicated $8^n \times 8^n$ banyans (with N sources). (c) *pb* for crossbar, 8-dilated $2^n \times 2^n$, 32-replicated $16^n \times 16^n$ and 4-dilated $4^n \times 4^n$ banyans (with N sources).

minimized. In SIMD systems, all processors must wait until each processor receives the data that it requested, and hence a number of memory cycles may be required. However, it appears that the required number of memory cycles will be minimized when the blocking probability under permutation requests is also minimized.

VI. Conclusions

We have presented analytic models for the blocking probability of crossbars and square multistage interconnection networks under the assumption of random memory requests. We have presented a technique of adjusting probabilities in these analytic models to yield approximations for the blocking probability under permutation requests. We have presented a technique to account for the first-order effects of statistical dependence in these approximations. However, we do not account for the effects of the data dependencies that arise in real applications.

We believe that the techniques used in this paper can be used to create analytic models for the blocking probability of arbitrary multistage interconnection networks, such as the ADM [3], GSN [8], ABN [17], under the assumption of permutation requests. Such analytic models quantify the permutation capability of these networks.

We have shown that multistage interconnection networks can be designed with very low blocking probabilities. Contrary to [16], it appears that dilations are far more cost effective than replications.

One possible application for interconnection networks with very low blocking probabilities under permutation requests is in supercomputers using data skewing algorithms. Analytic models for the steady-state blocking probability of multistage

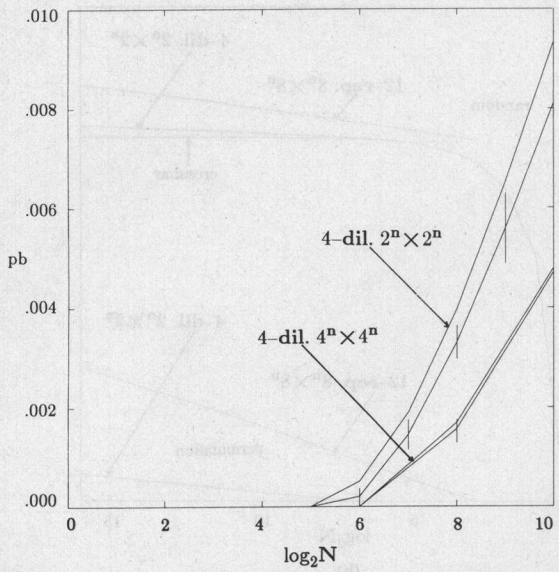


Fig. 8. *pb* for 4-dilated $2^n \times 2^n$ and 4-dilated $4^n \times 4^n$ banyans (with N sources).

interconnection networks used in computers using data skewing algorithms are still not available. Our analytic models do not account for data dependencies; further work in this area would be an extension of our model to account for dependencies.

References

[1] G. B. Adams, III, and H. J. Siegel, "The extra-stage cube: A fault-tolerant interconnection network for supersystems," *IEEE Trans. Comput.*, vol. C-31, pp. 443–454, May 1982.

[2] ——, "Modifications to improve the fault tolerance of the extra stage cube interconnection network," in *Proc. 1984 Int. Conf. Parallel Processing*, 1984, pp. 155–164.

[3] ——, "On the number of permutations performable by the augmented data manipulator," *IEEE Trans. Comput.*, vol. C-31, pp. 270–277, Apr. 1982.

[4] D. P. Agrawal, "Graph theoretical analysis and design of multistage interconnection networks," *IEEE Trans. Comput.*, vol. C-32, pp. 637–648, July 1983.

[5] G. H. Barnes and S. F. Lundstrom, "Design and validation of a connection network for many-processor multiprocessor systems," *IEEE Computer*, pp. 31–41, Dec. 1981.

[6] V. E. Benes, "On rearrangeable three-stage connecting networks," *Bell Syst. Tech. J.*, pp. 1481–1492, Sept. 1962.

[7] D. P. Bhandarkar, "Analysis of memory interference in multiprocessors," *IEEE Trans. Comput.*, vol. C-24, pp. 897–908, Sept. 1975.

[8] L. N. Bhuyan and D. P. Agrawal, "Design and performance of generalized interconnection networks," *IEEE Trans. Comput.*, vol. C-32, pp. 1081–1090, Dec. 1983.

[9] D. Y. Chang, D. J. Kuck, and D. H. Lawrie, "On the effective bandwidth of parallel memories," *IEEE Trans. Comput.*, vol. C-26, pp. 480–490, May 1977.

[10] C-Y. Chin and K. Hwang, "Connection principles for multipath packet switching networks," in *Proc. 11th Annu. Symp. Computer Architecture*, 1984, pp. 99–108.

[11] C. Clos, "A study of nonblocking switching networks," *Bell Syst. Tech. J.*, pp. 406–424, Mar. 1953.

[12] N. J. Davis and H. J. Siegel, "The performance analysis of partitioned circuit switched multistage interconnection networks," in *Proc. 12th Annu. Symp. Computer Architecture*, 1985, pp. 387–394.

[13] M. A. Franklin, "VLSI performance comparison of banyan and crossbar communications networks," *IEEE Trans. Comput.*, vol. C-30, pp. 283–290, Apr. 1981.

[14] M. A. Franklin, D. F. Wann, and W. J. Thomas, "Pin limitations of VLSI interconnection networks," *IEEE Trans. Comput.*, vol. C-31, pp. 1109–1116, 1982.

[15] G. R. Goke and G. J. Lipovski, "Banyan networks for partitioning multiprocessor systems," in *Proc. 1st Annu. Symp. Computer Architecture*, 1973, pp. 21–28.

[16] C. P. Kruskal and M. Snir, "The performance of multistage interconnection networks for multiprocessors," *IEEE Trans. Comput.*, vol. C-32, pp. 1091–1098, Dec. 1983.

[17] V. P. Kumar and S. M. Reddy, "Design and analysis of fault-tolerant multistage interconnection networks with low link complexity," in *Proc. 12th Annu. Symp. Computer Architecture*, 1985, pp. 376–386.

[18] T. Lang, "Interconnections between processors and memory modules using the shuffle-exchange network," *IEEE Trans. Comput.*, vol. C-25, pp. 496–503, May 1976.

[19] D. H. Lawrie, "Access and alignment of data in an array processor," *IEEE Trans. Comput.*, vol. C-24, pp. 1145–1155, Dec. 1975.

[20] ——, "The prime memory system for array access," *IEEE Trans. Comput.*, vol. C-31, pp. 435–442, May 1982.

[21] M. Lee and C-L Wu, "Performance analysis of circuit switching baseline interconnection networks," in *Proc. 11th Annu. Symp. Computer Architecture*, 1984, pp. 82–90.

[22] K. Padmanabhan and D. H. Lawrie, "A class of redundant path multistage interconnection networks," *IEEE Trans. Comput.*, vol. C-32, pp. 1099–1108, Dec. 1983.

[23] J. H. Patel, "Performance of processor-memory interconnections for multiprocessors," *IEEE Trans. Comput.*, vol. C-30, pp. 771–780, Oct. 1981.

[24] M. C. Pease, "The indirect binary n-cube microprocessor array," *IEEE Trans. Comput.*, vol. C-26, pp. 458–473, May 1977.

[25] C. V. Ravi, "On the bandwidth and interference in interleaved memory systems," *IEEE Trans. Comput.*, vol. C-21, pp. 899–901, Aug. 1972.

[26] S. M. Reddy and V. P. Kumar, "On fault-tolerant multistage interconnection networks," in *Proc. 1984 Int. Conf. Parallel Processing*, 1984, pp. 155–164.

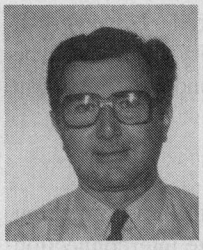
[27] H. J. Siegel, "Analysis techniques for SIMD machine interconnection networks and the effects of processor address masks," *IEEE Trans. Comput.*, vol. C-26, pp. 153–161, Feb. 1977.

[28] K. G. Shin and J-C Liu, "C-net: A cost-effective multistage interconnection network," Univ. Michigan Comput. Res. Lab. Tech. Rep. CRL-TR-47-84, 1984.

[29] N-F Tzeng, P-C Ye, and C-Q Zhu, "A fault-tolerant scheme for multistage interconnection networks," in *Proc. 12th Annu. Symp. Computer Architecture*, 1985, pp. 368–375.

Ted H. Szymanski received the B.A.Sc. degree in engineering science in 1980, and the M.A.Sc. degree in electrical engineering in 1982, from the University of Toronto, Toronto, Ont., Canada.

During 1983 he was a consultant to a number of firms in the Toronto area. Since 1984 he has been working on the Ph.D. degree at the University of Toronto. His current interests include computer architecture, VLSI, and applications of artificial intelligence to CAD.



V. Carl Hamacher (S'66–M'68–SM'79) was born in London, Canada, in 1939. He received the B.A.Sc. degree in engineering physics in 1963 from the University of Waterloo, Waterloo, Ont., Canada, the M.Sc. degree in electrical engineering in 1965 from Queen's University, Kingston, Ont., Canada, and the Ph.D. degree in electrical engineering in 1968 from Syracuse University, Syracuse, NY.

Since then he has been at the University of Toronto, Toronto, Ont., Canada, where he is a Professor in the Departments of Electrical Engineering and Computer Science, and the Director of the Computer Systems Research Institute. His current interests include local area computer networks and real-time computer systems. From 1978 to 1979, he was a Visiting Scientist at the IBM Research Laboratory in San Jose, CA. He is a coauthor of the textbook *Computer Organization* (New York: McGraw-Hill, 1984).

Dr. Hamacher is a member of the Association for Computing Machinery, Sigma Xi, and the Association of Professional Engineers of Ontario.