The Performance of Multistage Interconnection Networks for Multiprocessors

CLYDE P. KRUSKAL, MEMBER, IEEE, AND MARC SNIR

Abstract—**This paper studies the performance of unbuffered and buffered, packet-switching, multistage interconnection networks. We begin by reviewing the definition of banyan networks and introducing some generalizations of them. We then present an asymptotic analysis of the performance of unbuffered banyan networks, thereby solving a problem left open by Patel. We analyze the performance of the unbuffered generalized banyan networks, and compare networks with approximately equivalent hardware complexity. Finally, we analyze the performance of buffered banyan networks and again compare networks with approximately equivalent hardware complexity.**

Index Terms—**Bandwidth, banyan network, bidelta network, buffered network, circuit-switching network, crossbar network, delta network, dilated network, multistage interconnection network, packet-switching network, performance analysis, replicated network, simulation, square network, throughput, unbuffered network, uniform network.**

I. Introduction

WITHIN the last decade interest has increased in large scale multiprocessors composed of thousands of processors sharing a common memory, and several such systems have been proposed [5], [11]. A typical configuration for such a system is illustrated in Fig. 1: many identical processors are connected via an interconnection network to identical memory modules. The interconnection network is an essential component, so it is important to have a solid understanding of its performance. In some proposed designs it supports dynamic access from each processor to each memory module, and the traffic through the network consists of short items (requests to memory and replies), with requests being dynamically generated independently at each processor. The pattern of requests is essentially random and varies rapidly.

In this paper we study the performance of unbuffered and buffered, packet-switching, multistage interconnection networks. Section II reviews the definitions necessary for reading this paper, including specifically the definition of banyan networks and some generalizations of them. Sections III and V analyze the performance of banyan networks. Delta networks [9], Omega networks [8], indirect binary cube networks [11]—among others (see [7])—are all banyan networks, so our analyses apply to these networks as well. Patel [10] presents a recurrence relation for the performance of unbuffered networks, but leaves open the question of how they perform asymptotically. Section III provides such an analysis; the analysis is also applicable to circuit-switching networks. Section IV analyzes the performance of unbuffered generalized banyan networks, and compares networks with approximately equivalent hardware complexity. Section V analyzes the performance of buffered banyan networks, and as in the unbuffered case compares networks with approximately equivalent hardware complexity. Section VI summarizes the paper and contains some concluding remarks.

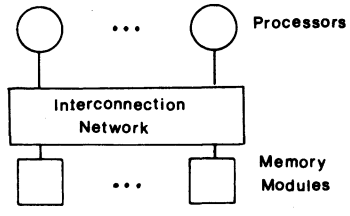


Fig. 1. Multiprocessor organization.

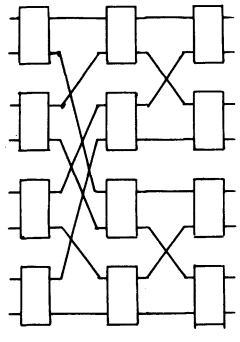


Fig. 2. 3-stage square banyan network of degree 2.

II. Networks

We consider packet-switching networks built of switches connected by unidirectional lines. A p input, q output ($p \times q$) switch can receive packets at each of its p input ports, and send them through each of its q output ports. A *network* is a directed graph where nodes are of the following three types:

Manuscript received May 21, 1982; revised October 29, 1982, March 2, 1983, and May 25, 1983. This work was supported in part by the Applied Mathematical Sciences Program of the U.S. Department of Energy under Contract DE-AC02-76ER03077, in part by the National Science Foundation under Grants NSF-MCS79-21258 and NSF-MCS81-05896, and in part by a grant from IBM. A preliminary version of this material appeared in the *Proceedings of the 1982 Conference on Information Sciences and Systems*, Princeton University, March 1982.

C. P. Kruskal is with the Department of Computer Science, University of Illinois, Urbana, IL 61801.

M. Snir was with Courant Institute, New York University, New York, NY 10012. He is now with the Department of Computer Science, Hebrew University of Jerusalem, Jerusalem, Israel.

0018-9340/83/1200-1091$01.00

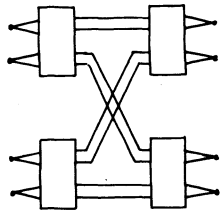


Fig. 3. 2-dilation of banyan network with 2 stages and degree 2.

i) *Source nodes* which have indegree 0;

ii) *Sink nodes* which have outdegree 0;

iii) *Switches* which have positive indegree and outdegree.

Each edge represents one of more lines going from a node to successor. Throughout this paper we assume that for a given network every edge represents the same number of lines. The edges in two different networks, however, may represent different numbers of lines.[1]

A *banyan network* is defined by Goke and Lipovsky [4] to be a network with a unique path from each source node to each sink node. This condition 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. A *multistage network* is a network in which the nodes can be arranged in stages, with all the source nodes at stage 0, and all the outputs at stage i connected to inputs at stage $i + 1$. If all the sink nodes of a multistage network are at stage $n + 1$ then we have an *n-stage network* (called in [4] L-level). A *uniform network* is a multistage network in which all switches at the same stage have the same number of input ports and the same number of output ports. A *square network of degree k* is network built of $k \times k$ switches. Fig. 2 shows a 3-stage square banyan network of degree 2.

We shall analyze the performance of n-stage square banyan networks under the assumptions usually used in the literature [10]: packets are generated at each source node by independent, identically distributed random processes. Each processor generates with probability p at each cycle a packet, and sends a generated packet with equal probability to any sink node. We assume that the network is synchronous, so that packets can be sent only at time $t_c, 2t_c, 3t_c, \cdots$, where t_c is the *network cycle time.*

The uniqueness of paths in banyan networks implies the following result which is implicitly used in all the performance analyses of these networks.

Lemma: Let packets be generated at the source nodes of a banyan network by independent, identically distributed random processes, that uniformly distribute the packets over all of the sink nodes. Assume that the routing logic at each switch is "fair," i.e., conflicts are randomly resolved. Then

i) The patterns of packet arrivals at the inputs of the same switch are independent.

ii) Packets arriving at an input of a switch are uniformly distributed over the outputs of that switch.

Moreover, if the network is uniform, then for each stage in the network, the pattern of packet arrivals at the inputs of that stage have the same distribution

While banyan networks are very attractive in their simplicity, other considerations such as performance or reliability sometimes dictate the use of more complicated networks. Two strategies can be used to augment a network G, without sacrificing much of its structure.

i) The d-dilation of G is defined to be the network obtained from G by replacing each edge by d distinct edges (see Fig. 3). A packet entering a switch may exit using any of the d edges going to the desired successor switch at the next stage.

ii) The d-replication of G is defined to be the network consisting of d identical distinct copies of G, with the d corresponding source (sink) nodes in each copy identified (see Fig. 4).

In [12] d-dilated networks are introduced without name. Replicated banyan networks are called layered banyan networks in [4].

III. Unbuffered Banyan Networks

We first consider packet-switching networks built of $k \times k$ unbuffered switches with the topology of an n-stage square banyan network. When several packets at the same switch require the same output, a randomly chosen one is forwarded and the remaining packets are deleted. The relevant figure of merit for such networks is the probability p_m that there is some packet on any particular input at the mth stage of the network. By the above lemma, it is easily seen that p_m is well defined and satisfies the recurrence relation

$$p_{m+1} = 1 - (1 - p_m/k)^k \tag{1}$$

with boundary condition $p_0 = p$, where p is the probability of packet creation at a source node. Patel [9] leaves open the question of how p_m behaves asymptotically. It turns out that for any fixed initial value $p_0 > 0$, and any fixed k

$$p_m = \frac{2k}{(k-1)m}\left[1 - \frac{(k+1)}{3(k-1)}\frac{\ln m}{m} + O\left(\frac{1}{m}\right)\right], \tag{2}$$

where $\ln x$ denotes $\log_e x$. A proof of this result is given in Appendix A. It is interesting to note that the first and second order terms in this expansion are independent of p_0. Thus, the probability that a message is not deleted is asymptotically inversely proportional to the number of stages in the network.

[1] This definition allows us to study the performance of unidirectional interconnection networks. However, in actual parallel computers of the form described in Fig. 1, it is necessary to send replies back through the network. There are at least four ways to reinterpret or modify our definition to allow this. Assume that the parallel processor has P processors and M memory modules.

i) Each edge really represents a bidirectional line. There are P source nodes and M sink nodes; the orientation of the edges distinguishes processors from memory modules.

ii) Each edge really represents two unidirectional lines—one in each direction. Again, there are P source nodes and M sink nodes, and the orientation of the edges distinguishes processors from memory modules.

iii) There is one network for sending messages from processors to memory modules and a different network for sending replies back. In one network each of the P source nodes represents a processor, and each of the M sink nodes represents a memory module; in the other network each of the P sink node represents a processor and each of the M source node a memory module.

iv) The same (unidirectional) network is used for sending messages and replies. There are $P + M$ source nodes and $P + M$ sink nodes. Each source node is identified with a distinct sink node, and the pair represents either a processor or a memory module.

In each of these cases it is not hard to at least approximate the behavior the full network given our analyses below of unidirectional networks.

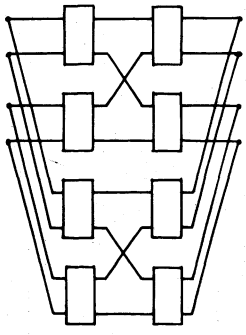


Fig. 4. 2-replication of banyan network with 2 stages and degree 2.

In particular, in a square banyan network with N source nodes built of 2×2 switches the *bandwidth* (or *throughput*) of the network, i.e., the average number of packets arriving at the other end of the network per cycle, is

$$N \cdot p_{\lg N} = \frac{4N}{\lg N}\left[1 - \frac{\ln \lg N}{\lg N} + O\left(\frac{1}{\lg N}\right)\right],$$

where $\lg x$ denotes $\log_2 x$.

We can use (2) to obtain for any given $p = p_0$ an approximate formula for p_m. Although we do this only for $k = 2$, the technique is valid in general. For $k = 2$ the formula for p_m reduces to

$$\frac{4}{m}\left[1 - \frac{\ln m}{m} + O\left(\frac{1}{m}\right)\right].$$

Assuming the last term does not oscillate (which can be shown formally by generating the next term in the asymptotic expansion), p_m can be approximated by

$$\frac{4}{m}\left[1 - \frac{\ln m}{m} + \frac{c}{m}\right],$$

where c is some constant to be determined. We can determine a value for c by taking a "large" value for m, finding p_m by brute force using difference equation (1) m times, and solving for c. For example, consider $p = 1$. For $m = 10$, $p_0 = 1$, $p_1 = 0.75$, $p_2 \simeq 0.61, \cdots, p_{10} \simeq 0.26$, so $c \simeq -1.2$. Thus, for m large, p_m is closely approximated by

$$\frac{4}{m}\left[1 - \frac{\ln m}{m} - \frac{1.2}{m}\right]. \tag{3}$$

Unfortunately, for each initial value p we need a different constant c. Moreover, for small p the approximation will not be consistently good until m is very large, no matter what value we choose for c. Using an alternative approach we can derive an asymptotic formula which has p as a parameter.

Let p_m satisfy the recurrence relation with initial value $p_0 = p$, and let $f(x)$ (defined on the real numbers) approximate p_m for some initial value greater than or equal to p. Then f, starting at the point where it has value p, approximates p_m; that point is by definition $(f^{-1}(p), p)$. Thus, $f(f^{-1}(p) + m)$ approximates p_m (see Fig. 5). For our f we use only the first term $4/m$ of the above asymptotic formula for two reasons: first of all, we can invert it in closed form; and second, its range contains the entire half open interval (0, 1]. Thus $f^{-1}(p) = \frac{4}{p}$, and the approximation for p_m is $f(f^{-1}(p) + m) = f\left(\frac{4}{p} + m\right)$

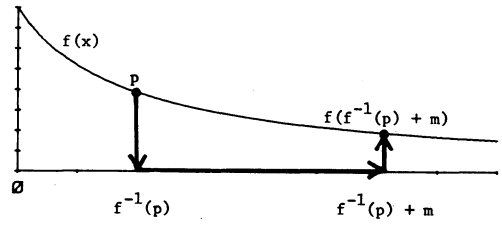


Fig. 5. Schema for finding asymptotic equation approximating p_m. The arrows show the mapping from the point $(f^{-1}(p), p)$ to the approximation of p_m. The left axis is nominally a probability, but since f only approximates p_m the range of the graph can contain values greater than 1.

$$= \frac{4}{m + \frac{4}{p}}. \tag{4}$$

This is an excellent approximation of p_m for all p and m, except when both p is large and m is small where it is still very good. (See Fig. 6.) In general, for square networks composed of $k \times k$ switches p_m is approximated by

$$\frac{2k}{(k-1)m + \frac{2k}{p}}.$$

The above analysis also applies to a circuit-switching network with the topology of a square banyan. We assume that each source node attempts simultaneously to establish a communication path with a randomly chosen sink node (each mapping is equally likely). Then p_m is the probability that a communication is not blocked after the first m stages of the network, and $N \cdot p_{\lg N}$ is the average number of communication paths that will be established in a network with N source nodes built of 2×2 switches. Our asymptotic analysis indicates that the probability that a path will be established is asymptotically inversely proportional to the number of stages in the network.

IV. Unbuffered Dilated and Replicated Networks

We now analyze the performance of d-dilated square banyan networks under the following two assumptions: 1) Every source node issues one message at each cycle. 2) If $m < d$ packets are competing for the d edges leading from a switch to its successor switch at the next stage, all of the packets are forwarded; if $m > d$ packets are competing at a switch for the d edges, then d of them are chosen at random and forwarded, and the remaining ones are deleted. We give the following recurrence for q_m, the probability that the d edges will contain some message after the first m stages of a d-dilated square banyan network of degree 2.

Let $R(m, j)$ be the probability that j packets are transmitted through d identical edges leaving a switch at stage m. R is initialized as

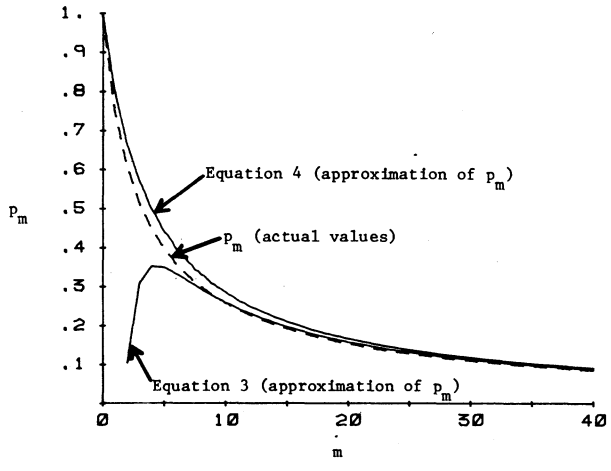


Fig. 6. Graph of p_m, for $p_0 = 1$, and two asymptotic equations for finding approximate values.

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

$$R(0, j) = 1, \text{ for } j = 1.$$

The probability of having i messages entering a switch at stage $m + 1$ is given by $\sum_{r+s=i} R(m, r) \cdot R(m, s)$. The probability that j of these i messages are directed to a fixed output port is $\binom{i}{j} 2^{-i}$. Thus, $R(m + 1, j)$ is given by

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

and

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

Finally, the probability that at least one of the d edges contains a message is

$$q_m = 1 - R(m, 0).$$

The performance of d-replicated square banyan networks can be easily approximated. A d-replicated network consists of d copies of a square banyan networks, and the results of Section III apply to each of these copies. We assume that every source issues one message at each cycle and randomly sends it to one of the d copies. On each copy the probability p_m that a message survives m stages of the network satisfies the recurrence from equation (1). Assuming the d copies are independent, which is slightly optimistic, the probability q_m that the ith edge of one of the d copies has a surviving request is $1 - (1 - p_m)^d$. Our numerical results assume this model of d-replication. Note, however, that if k divides d, a d-replicated square banyan network of degree k could be organized so that there will be no conflicts until after $\log_k d$ stages of switches. The analysis of each copy is equivalent to assuming each banyan network has $\log_k N - \log_k d$ stages, where the probability is still $1/d$ of issuing a request at each input port. Now the d copies are independent, so this organization is easy to analyze; it performs better than even the above optimistic approximation assumes.

A d-dilation of a square banyan network of degree 2 with N source nodes has $\lg N$ stages, $(N \lg N)/2$ $2d \times 2d$ switches, and $dN(\lg N + 1)$ edges. A $d(\lg d + 1)$-replication of a square banyan network of degree $2d$ has $\log_{2d} N$ stages, $(N \lg N)/2$ $2d \times 2d$ switches, and $dN(\lg N + \lg d + 1)$ edges. Thus two such networks have the same number and size of switches, and have roughly the same number of edges. Figs. 7(a)-(c) compares the performance—as measured by q_m—of dilated banyan networks with replicated banyan networks of comparable complexity for switches of size 4×4, 8×8, and 16×16. We have also included the crossbar network for comparison purposes. (In a crossbar network of size N the probability that a message is not blocked is $1 - (1 - 1/N)^N$, which asymptotically approaches $1 - 1/e$.) As we see dilated banyan networks have asymptotically better performance than replicated banyan networks, although for practical values of N their performances are approximately the same.

We have also calculated the bandwidth of these networks. To do so we assume that a message is issued at each input edge at each cycle. For d-dilated networks the recurrence is the same as above except for different boundary conditions and a different formula for calculating q_m. In this case

$$R(O, j) = 0, \text{ for } j \neq d,$$

$$R(O, j) = 1, \text{ for } j = d,$$

and

$$q_m = \frac{1}{d} \sum j \cdot R(m, j).$$

For d-replicated networks the bandwidth is exactly d times the bandwidth of a single network. Once again dilated networks have asymptotically better performance: This occurs, however, only for impractically large values of N: for $N < 2^{60}$ replicated networks built of 4×4, 8×8, or 16×16 switches are better than comparable dilated networks.

V. Buffered Banyan Networks

The bandwidth of packet-switching networks can be improved by using buffers to queue conflicting packets. An accurate analysis of the performance of buffered square banyan networks does not seem tractable. Several authors have analyzed the performance with buffers of length one and performed simulations for larger buffers (see [3] and references therein, and more recently [1]). We present here a formula that seems to yield a good approximation for the performance with large buffers and also indicate some tradeoffs suggested by that formula.

Consider a buffered $k \times k$ switch. An ideal switch consists of k infinite queues, one associated with each output port, where each queue can accept at each cycle up to k distinct packets coming from distinct input ports. In practice a switch can be built in the following way. To prevent blocking each such queue is actually implemented by k FIFO buffers, one

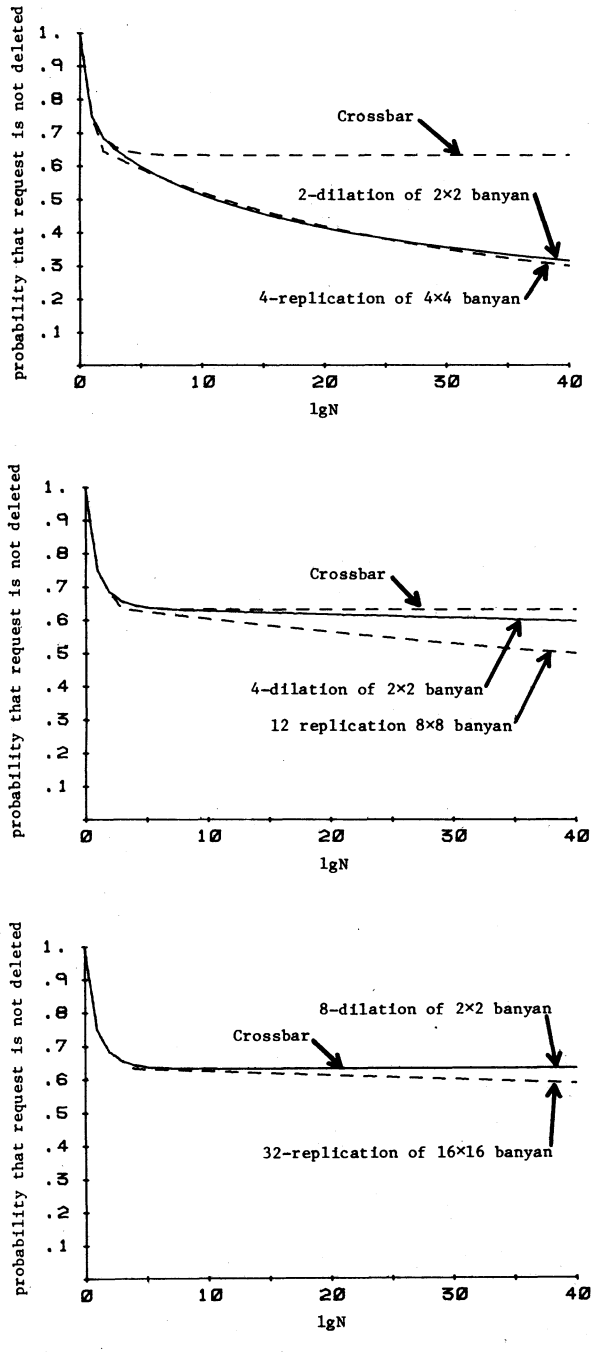


Fig. 7. (a) Probability of a request being satisfied in a network with N source nodes using a crossbar, 2-dilation of 2 × 2 network, and 4-replication of 4 × 4 network. (b) Probability of a request being satisfied in a network with N source nodes using a crossbar, 4-dilation of 2 × 2 network, and 12-replication of 8 × 8 network. (c) Probability of a request being satisfied in a network with N source nodes using a crossbar, 8-dilation of 2 × 2 network, and 32-replication of 16 × 16 network.

at each input port. Incoming packets are therefore directly entered into a buffer associated with the destination output port, with no conflicts arising. An arbitration mechanism is used at each output port to remove packets from the k associated buffers in the order of their arrivals (see [13] for an implementation of such switch). (In the terminology of Dias and Jump [3] we have $t\_\_pass = 0$.) Assuming that the buffers are infinite, each switch gives the same performance as an ideal switch.

We now analyze the performance of networks composed of ideal switches. Let t_c be the *cycle time* of the switch, i.e., the interval between successive packet arrivals; let t_τ be the *transit time* of a packet from one switch to the next one when the buffers on its path are empty ($t_\tau \le t_c$). In general, the transit time of a packet P through a switch is $t_\tau + bt_c$, where b is the number of packets with the same destination that arrived before P, or arrived at the same time as P, but were transmitted before P (we assume that the order of transmission of packets arriving at the same cycle is random). If at each cycle a packet arrives on each input with probability p, then the average number of queueing cycles for a packet is

$$b_{\text{av}} = \frac{(1 - 1/k)p}{2(1 - p)}.$$

A proof of this formula is given in Appendix B. (Chen [2] has independently derived this formula for the case $k = 2$.) Thus, the average transit time of a packet through a $k \times k$ switch is

$$t_k = t_\tau + t_c \frac{(1 - 1/k)p}{2(1 - p)}.$$

This suggests a formula of the form

$$T = \log_k N \cdot t_k$$
$$= \log_k N \cdot \left(t_\tau + t_c \frac{(1 - 1/k)p}{2(1 - p)}\right)$$

for the average transit time through the $\log_k N$ stages of a square banyan network of degree k with N source nodes.

Simulations were run on six stages of a network built of 2 × 2 switches. The assumption of infinite buffers was dropped: each switch was given a buffer of size eight at each output port. Table I compares the average number of queueing cycles predicted by this analysis with values obtained by the simulation. (The "packets per cycle" row of the simulation section shows the average number of packets actually generated per cycle; only the last column differed from the expected number.)

As we see, the predicted delays are in good agreement with the simulations. Besides statistical error, there are two reasons for the discrepancies between the two. First, buffers have only finite size, and a full switch will not accept a new packet. However, the close agreement between the number of transmissions predicted and simulated indicates this does not occur frequently, so that limited buffer size does not seem to be a significant factor for the loads considered. Second, the distribution of the packet arrivals at the second and the subsequent stages is not time independent anymore. A clustering effect occurs, which tends to increase the average delays. Indeed, delays can be seen to increase from the first stage to the successive stages. It is interesting to note that for each p after the second stage there is no discernible difference between the delay times. This is probably because the distribution of packet arrivals has by that time pretty much settled down to its limiting distribution.

Sometimes each packet will not represent a full message, but a message will be composed of several packets, say m. In this case each message is said to be *time multiplexed* by a

TABLE I

probability of transmission			0.2	0.4	0.6	0.8
analysis	packets per cycle		0.2	0.4	0.6	0.8
	waiting per stage		0.063	0.167	0.375	1.265
simulation	packets per cycle		0.200	0.400	0.600	0.795
	waiting time	1st stage	0.066	0.167	0.367	1.082
		2nd stage	0.065	0.175	0.434	1.275
		3rd stage	0.069	0.201	0.457	1.328
		4th stage	0.069	0.195	0.456	1.316
		5th stage	0.070	0.202	0.431	1.298
		6th stage	0.066	0.196	0.459	1.289

factor m. If the cycle time for to process a single packet is t_c and the transit time is t_τ, the cycle time to process a message that is time multiplexed by a factor m will be $T_c = mt_c$, whereas the transit time will be unchanged. The average number of packets per cycle will now be $P = mp$. Using the previous formula, one obtains that the average transit time through a network (built of $k \times k$ switches in which messages are time multiplexed by a factor m) is approximately

$$T = \log_k N \cdot \left(t_\tau + T_c \frac{(1 - 1/k)P}{2(1 - P)}\right) + (T_c - t_c)$$

$$= \log_k N \cdot \left(t_\tau + t_c \frac{m^2(1 - 1/k)p}{2(1 - mp)}\right) + (m - 1)t_c.$$

The last term accounts for the pipe setting delay.

Furthermore, we can consider d-replicated square banyan networks. Recall that a d-replicated banyan network consists of d copies of a banyan network. When a processor issues a message it sends the message to one of the d copies with equal probability. Thus, the average number of packets per cycle on each copy is now p/d. So, ignoring the overhead in connecting the d copies together, the average transit time through a d-replicated square banyan network is approximately

$$T = \log_k N \cdot \left(t_\tau + t_c \frac{m^2(1 - 1/k)p}{2(d - mp)}\right) + (m - 1)t_c.$$

It is possible to build square banyan networks with different performances by increasing the number of ports on each switch while proportionally decreasing the bandwidth of each port (i.e., the number of lines per edge). For VLSI chips if the computation time is small enough relative to the I/O requirements then the chip is said to be pin limited (i.e., the time to compute the desired function mainly depends on how many pins are allocated to its I/O, and is inversely proportional to this number). If each switch is implemented on one chip and the chip performance is pin limited, then without changing the transit time the number of logical lines per switch can be increased by a factor of l while increasing the multiplexing factor of each message by the by the same factor l. Assuming that a 2×2 switch can process each message as one packet, a $k \times k$ switch will require $k/2$ packets per message. Thus the average transit time through a $k \times k$ square banyan network will be approximately

$$T = \log_k N \cdot \left(t_\tau + t_c \frac{k(k - 1)p}{8(1 - kp/2)}\right) + \left(\frac{k}{2} - 1\right)t_c.$$

We have used this formula to compare the bandwidth of networks built of $k \times k$ switches, $k = 2, 4, 8, \cdots$, assuming that $t_c = t_\tau$. Fig. 8 indicates the domain where each type of switch yields the best performance. Larger switches perform better for low traffic intensities and large networks.

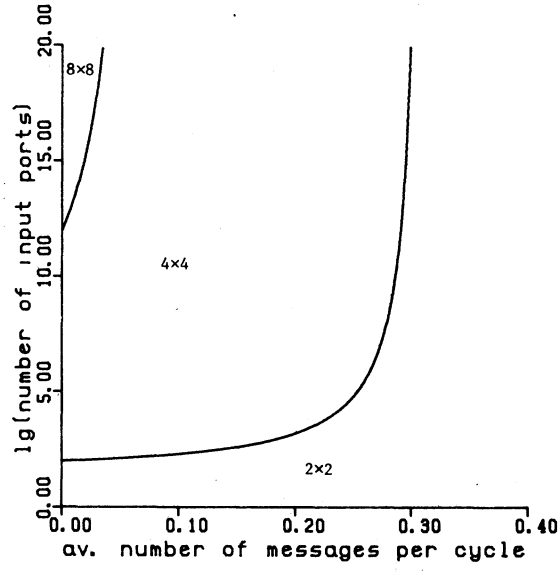


Fig. 8. Domain of best performance for a square banyan network of degree 2^k, $k = 1, 2, \cdots$.

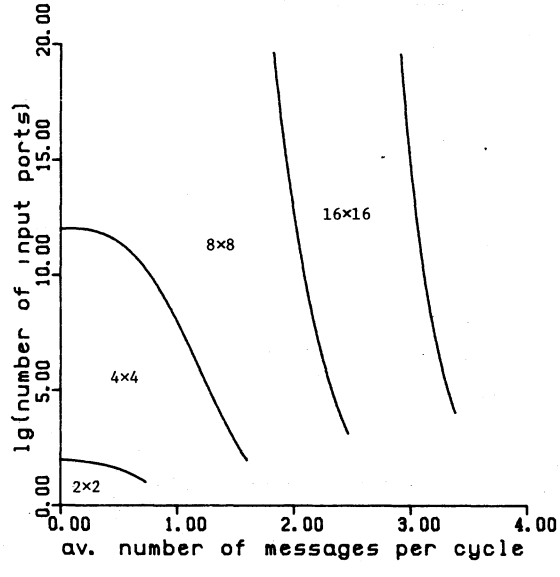


Fig. 9. Domain of best performance for replicated square banyan networks of degree 2^k, $k = 1, 2, \cdots$, of comparable complexity.

Note, however, that a network built of 2×2 switches has the same number of switches as a $(k \lg k)/2$-replication of a network built of $k \times k$ switches. Furthermore, assuming each edge of a 2×2 switch is given $k/2$ times as many lines per edge as a $k \times k$ switch, the different networks will have the same number of lines per switch, and networks with identical numbers of switches will have roughly the same total number of lines. So two such networks will have comparable complexity. Comparing the performance of networks of comparable complexity gives another picture, which is illustrated in Fig. 9. Networks with larger switches are capable of supporting higher traffic intensities. In particular, four networks built of 4×4 switches have the same number of switches, and always outperform one network built of 2×2 switches.

VI. Summary and Conclusion

We have studied the performance of unbuffered and buffered multistage interconnection networks. For unbuffered banyan networks we presented an asymptotic equation for the probability that a request issued at a source node arrives at its sink node. From the analysis we saw that asymptotically this probability is inversely proportional to the number of stages in the network. Furthermore, using the analysis we derived very simple equations which closely approximate the actual probabilities.

We then analyzed the performance of unbuffered dilated networks and unbuffered replicated networks, and compared networks of comparable hardware complexity. Although dilated networks provide asymptotically better performance, for practical numbers of processors dilated and replicated networks have similar performance. Thus, other considerations such as delay, fault tolerance, and layout would likely decide which is preferable for a given situation.

Finally, we derived an equation for the performance of buffered banyan networks. While the analysis is valid only for the first stage, we have presented evidence that it is a reasonable approximation in general. Using the analysis, we compared buffered banyan networks built of different sized switches but comparable hardware complexity, and determined where each switch size is most effective.

All of our analyses were done in the framework of the banyan network since it is the most general class for which the analyses apply. In practice, however, for ease of communication between sources and sinks one would use delta networks [9] where the paths from the source nodes to any specific sink node have the same descriptor, or more likely bidelta networks [7] where also the paths from the sink nodes to any specific source node have the same descriptor.

Appendix A

We derive an asymptotic estimate for the sequence defined by the recurrence $p_{n+1} = 1 - (1 - p_n/k)^k$, where $0 < p_0 \leq 1$.

It is easy to check that the sequence p_n is monotonically decreasing to zero. Let $r_n = 1/p_n$. The sequence r_n is monotonically increasing to infinity, and fulfills the recurrence

$$\frac{1}{r_{n+1}} = 1 - \left(1 - \frac{1}{kr_n}\right)^k = \frac{1}{r_n} - \frac{k-1}{2kr_n^2} + \frac{(k-1)(k-2)}{6k^2r_n^3} + O\left(\frac{1}{r_n^4}\right).$$

It follows that

$$r_{n+1} = r_n + \frac{k-1}{2k} + \frac{k^2-1}{12k^2 r_n} + O\left(\frac{1}{r_n^2}\right). \tag{A.1}$$

Thus, for n large enough,

$$r_{n+1} > r_n + \frac{k-1}{2k}.$$

Summing for n we obtain the inequality

$$r_n \geq \frac{k-1}{2k} \cdot n + O(1).$$

Substituting back into (A.1) we obtain the inequality

$$r_{n+1} \leq r_n + \frac{k-1}{2k} + \frac{k^2-1}{12k^2\left(\frac{k-1}{2k} \cdot n + O(1)\right)} + O\left(\frac{1}{n^2}\right) = r_n + \frac{k-1}{2k} + \frac{k+1}{6kn} + O\left(\frac{1}{n^2}\right).$$

Summing again we obtain

$$r_n \leq \frac{k-1}{2k} \cdot n + \frac{k+1}{6k} \cdot \ln(n) + O(1). \tag{A.2}$$

Substituting again into (A.1) we obtain

$$r_{n+1} \geq r_n + \frac{k-1}{2k} + \frac{k^2-1}{12k^2\left(\frac{k-1}{2k} \cdot n + \frac{k+1}{6k} \cdot \ln(n) + O(1)\right)} + O\left(\frac{1}{n^2}\right) = r_n + \frac{k-1}{2k} + \frac{k+1}{6kn} + O\left(\frac{\ln(n)}{n^2}\right).$$

Summing a final time we obtain

$$r_n \geq \frac{k-1}{2k} \cdot n + \frac{k+1}{6k} \cdot \ln(n) + O(1). \tag{A.3}$$

It follows from inequalities (A.2) and (A.3) that

$$r_n = \frac{k-1}{2k} \cdot n + \frac{k+1}{6k} \cdot \ln(n) + O(1),$$

so that

$$p_n = \frac{1}{r_n} = \frac{1}{\frac{k-1}{2k} \cdot n + \frac{k+1}{6k} \cdot \ln(n) + O(1)} = \frac{2k}{(k-1)n} \cdot \left(1 - \frac{k+1}{3(k-1)} \cdot \frac{\ln(n)}{n} + O\left(\frac{1}{n}\right)\right).$$

Appendix B

We derive the average waiting time in a $k \times k$ switch with infinite buffers. We assume that a queue of unbounded capacity is associated with each output port, and that at each cycle each such queue can accept up to k packets coming through k distinct input ports. At each cycle a packet arrives at each input port with a fixed probability p, and each packet is equally likely to join each output queue. We take the cycle time t_c to be equal to 1.

Let v_n be the number of packets joining a fixed output queue at cycle n. Then, $v_1, v_2, \cdots$, are independent random variables

with a $b(\cdot; k; p/k)$ Bernoulli distribution. The expected number of arrivals is $E = k \cdot (p/k) = p$ and the variance is $V = k \cdot (p/k) \cdot (1 - p/k) = p(1 - p/k)$. Let q_n be the number of packets in the queue at the end of cycle n. We have for q_n the recurrence relation

$$q_{n+1} = q_n + v_{n+1} - 1 \quad \text{if } q_n > 0,$$
$$q_{n+1} = v_{n+1} \quad \text{if } q_n = 0.$$

Note that this recurrence is formally identical to that describing the number of customers at departure times in a $M/G/1$ queueing system [6, eq. (5.33)]. We can therefore use the derivation that leads to the Pollaczek-Khinchine mean-value formula to obtain the expected number of packets in the queue to be

$$\bar{q} = \frac{E}{2} + \frac{V}{2(1 - E)}.$$

Using Little's identity we get that the average system time for a packet is

$$\bar{s} = \frac{\bar{q}}{E} = \frac{1}{2} + \frac{V}{2E(1 - E)},$$

and the average waiting time is

$$\bar{w} = \bar{s} - 1 = \frac{V}{2E(1 - E)} - \frac{1}{2}.$$

Substituting for E and V we get that the average number of queueing cycles at a $k \times k$ switch is equal to

$$\bar{w} = \frac{p(1 - p/k)}{2p(1 - p)} - \frac{1}{2}$$
$$= \frac{(1 - 1/k)p}{2(1 - p)}.$$

ACKNOWLEDGMENT

We would like to thank the referees for carefully reading earlier versions of this paper. Their suggestions improved both the technical contents and presentation.

REFERENCES

[1] S. Cheemalavagu and M. Malek, "Analysis and simulation of banyan interconnection networks with 2 × 2, 4 × 4, and 8 × 8 switching elements," in *Proc. Real-Time Syst. Symp.*, Dec. 1982, pp. 83–89.

[2] P.-Y. Chen, "Multiprocessor systems: interconnection networks, memory hierarchy, modeling, and simulations," Ph.D. dissertation, Univ. Illinois, Urbana, IL, Jan. 1982.

[3] D. M. Dias and J. R. Jump, "Packet switching interconnection networks for modular systems," *Computer*, vol. 14, pp. 43–54, Dec. 1981.

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

[5] A. Gottlieb, R. Grishman, C. P. Kruskal, K. P. McAuliffe, L. Rudolph, and M. Snir, "The NYU Ultracomputer–Designing an MIMD, shared-memory parallel machine," *IEEE Trans. Comput.*, vol. C-32, pp. 175–189, 1983.

[6] L. Kleinrock, *Queueing Systems, Vol 1: Theory*. New York: Wiley, 1975.

[7] C. P. Kruskal and M. Snir, "The structure of multistage interconnection networks for multiprocessors," manuscript; see also, "Some results on multistage interconnection networks for multiprocessors," NYU Ultracomputer Note 41, in *Proc. 1982 Conf. Informat. Sci., Syst.*, Princeton Univ., Princeton, NJ, Mar. 1982.

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

[9] J. A. Patel, "Processor-memory interconnections for multiprocessors," in *Proc. 6th Annu. Symp. Comput. Arch.*, pp. 168–177, Apr. 1979.

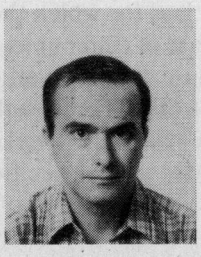
[10] ——, "Performance of processor-memory interconnections for multiprocessors," *IEEE Trans. Comput.*, vol. C-30, pp. 771–780, 1981.

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

[12] J. T. Schwartz, "The Burroughs FMP machine," Ultracomputer Note 5, Courant Institute, NYU, New York, NY, 1980.

[13] M. Snir and J. Solworth, "The Ultraswitch—A VLSI network node for parallel processing," Ultracomputer Note 39, Courant Institute, NYU, New York, NY, 1982.

Clyde P. Kruskal (M'83), for a photograph and biography, see p. 946 of the October 1983 issue of this TRANSACTIONS.



Marc Snir was born in Paris, France, in 1948. He received the B.Sc. degree in 1972, and the Ph.D degree in 1979, both in mathematics, from the Hebrew University of Jerusalem, Jerusalem, Israel.

From 1979 to 1980 he was a Research Fellow at the University of Edinburgh, Edinburgh, Scotland. From 1980 to 1982 he was an Assistant Professor at New York University, New York, where he was involved in the Ultracomputer research project.

Since 1982 he has been a Senior Lecturer in the Department of Computer Science, at the Hebrew University. His research interests include parallel processing, communication networks, and algorithmic complexity.

Dr. Snir is a member of the Association for Computing Machinery and the IEEE Computer Society.