Tinklepaugh
4E09

Performance of Multiprocessor Interconnection Networks

Laxmi N. Bhuyan, University of Southwestern Louisiana
Qing Yang, University of Rhode Island
Dharma P. Agrawal, North Carolina State University

With device characteristics approaching physical limits, parallel or distributed processing has been widely advocated as a promising approach for building high performance computing systems. The continued impetus in research in these areas arises from two factors: (a) the technological development in the area of VLSI chips and (b) the observation that significant exploitable software parallelism is inherent in many scientific and engineering applications.

To exploit this parallelism efficiently, a parallel/distributed system must be designed to considerably reduce the communication overhead between the processors. The communication architecture of the system might support one application well but might prove inefficient for others.

Therefore, we need to take a general approach, independent of the application, while designing the communication system or the interconnection network (IN) of a general-purpose parallel/distributed system. The IN must be efficient, reliable, and cost effective. A complete interconnection, such as a crossbar, might be cost prohibitive, but a shared-bus interconnection might be inefficient and unreliable. Thus, present research is directed to designing INs whose cost and performance lie somewhere between the two extremes.

Multiprocessor designers need analytical techniques to evaluate network performance. This article presents a tutorial on these evaluation tools to guide designers through the design process.

Ongoing research in the area of parallel and distributed processing suggests a number of promising INs. Because of the high cost involved in hardware implementation or software simulation of these INs, performance evaluation of these networks needs to be carried out through analytical techniques so that we can make a choice between various alternatives. A mathematical model makes it possible to study the efficiency of the IN in terms of various design parameters used as inputs to a model. Therefore, the intent of this article is to provide a tutorial on the subject of performance evaluation of multiprocessor interconnection networks to guide system designers in their design process.

A classification of parallel/distributed systems. We can divide general-purpose parallel/distributed computer systems into two categories: multiprocessors and multicomputers. The main difference between them lies in the level at which interactions between the processors occur.

A multiprocessor must permit all processors to directly share the main memory. All the processors address a common main memory space. In a multicomputer, however, each processor has its own memory space, and sharing between the processors occurs at a higher level as with a complete file or data set. A processor cannot directly access another processor's local memory.

Multiprocessors can be further divided as tightly coupled and loosely coupled. In a tightly coupled system, the main memory is situated at a central location so that the access time from any processor to the

0018-9162/89/0200-0025$01.00©1989 IEEE

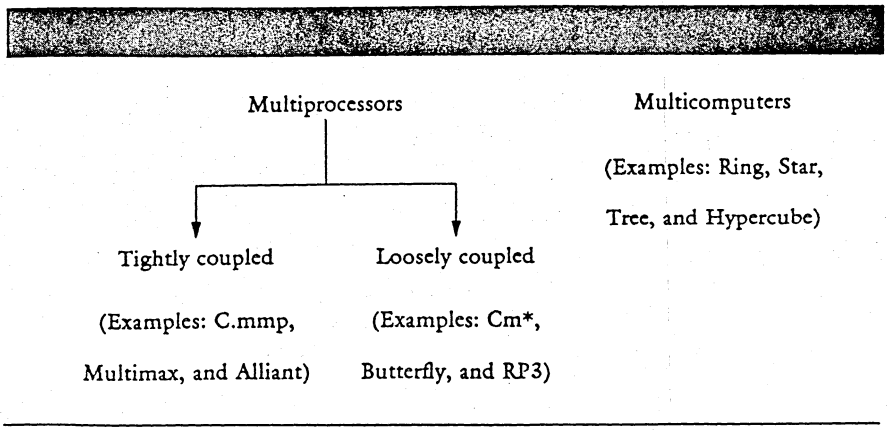


Figure 1. A classification of parallel/distributed systems.

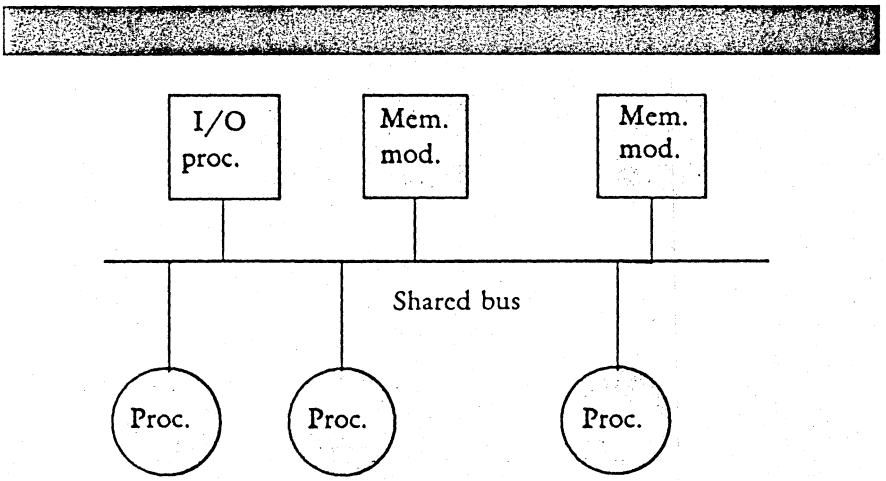


Figure 2. A single shared bus structure.

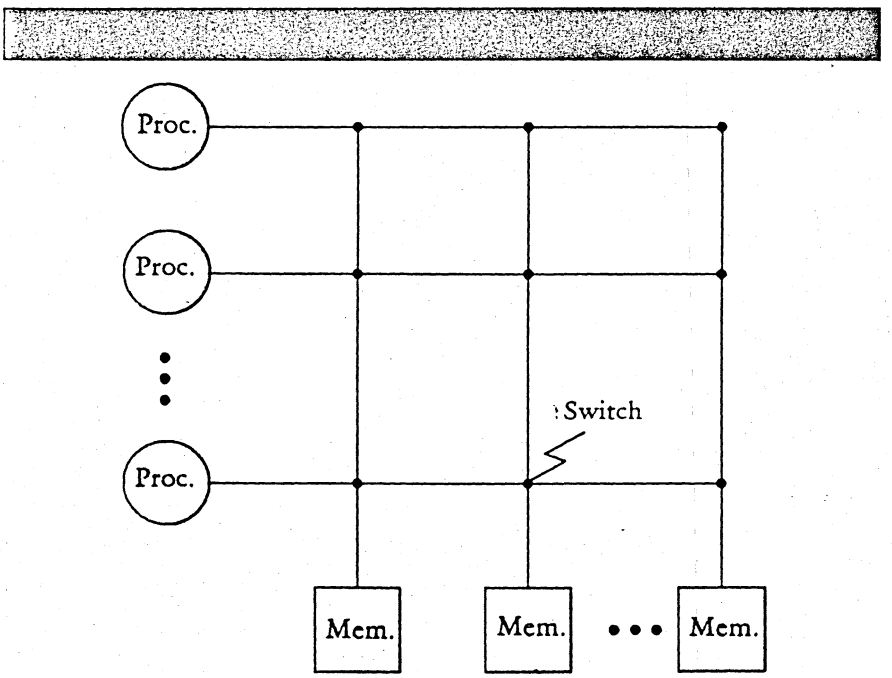


Figure 3. A crossbar interconnection network.

memory is the same. In addition to this central memory (also called main memory, shared memory, global memory, etc.), each processor might consist of some local memory or cache. The C.mmp of Carnegie Mellon University, the Multimax of Encore Computer, the FX of Alliant, and the Balance series of Sequent Corp. are examples of such tightly coupled multiprocessors.

In a loosely coupled system, the main memory is partitioned and attached to the processors, although the processors share the same memory address space. A processor can directly address a remote memory, but the access time is much higher compared to a local memory access. As a result, partitioning and allocation of program segments and data play a crucial role in the overall performance of an application program. The Cm* of CMU, the Butterfly machine of BBN Laboratories, and the RP3 of IBM are examples of such architectures.

As mentioned previously, the memory in a multicomputer is not shared. The interaction between the processors relies on message passing between the source and destination processors (nodes). The message passes over a link that directly connects two nodes and might have to pass through several such nodes in a store-and-forward manner before it reaches its destination. Therefore, each interaction involves a lot of communication overhead, and only those applications that need less interprocessor communication are well suited to multicomputers.

The multicomputers are usually based on topologies such as ring, tree, star, hypercube, etc. Hypercube machines such as Intel's iPSC are commercially available. Based on the description above, a classification of parallel/distributed computers appears in Figure 1. The classification does not include array and pipelined computers and local area networks. This is because array or pipelined computers are part of parallel processing but not distributed processing and, similarly, LANs are part of distributed processing but not parallel processing.

Essentially, our classification is valid for multiple instruction stream, multiple data stream computers. This article will concentrate solely on multiprocessor INs. A discussion of the performance of multicomputer interconnection networks can be found in Reed and Grunwald.[1]

Multiprocessor IN topologies. A multiprocessor organization is defined in terms

of the IN used. The performance of a multiprocessor rests primarily on the design of its IN. A shared-bus interconnection, shown in Figure 2, is the least complex and most popular among manufacturers. The Multimax and Alliant are examples of such multiprocessors. The shared bus does not allow more than one transfer between the processors and memories at a time. A large number of processors means a long wait for the bus.

On the other hand, a crossbar, as used in C.mmp and depicted in Figure 3, supports all possible distinct connections between the processors and memories simultaneously. Unfortunately, the cost of such a network is $O(NM)$ for connecting N inputs and M outputs. For a system with hundreds of processors and memories, the cost of such an IN is prohibitively high.

In terms of cost and performance, multistage interconnection networks (MINs) and multiple-bus networks achieve a reasonable balance between those of a shared bus and crossbar. MINs and multiple-bus networks are depicted in Figures 4 and 5, respectively, and will be described in later sections. Then, we will investigate the performance of these networks. A shared bus is essentially a special type of multiple-bus IN with the number of buses equal to one.

Classification of INs

An IN is a complex connection of switches and links that permits data communication between the processors and memories. Depending on the timing philosophy, switching mode, and control strategy, each set of topologically equivalent networks can have different operational characteristics giving rise to different system behaviors. These operational characteristics also necessitate different methodologies to be used in IN performance evaluation.

Timing philosophy. Timing philosophy is one of the most important attributes characterizing a communication system. Basically, there are two types of possible timing schemes in a system: synchronous and asynchronous.

Synchronous control techniques are well understood and widely used in computer system designs. They are characterized by the existence of a central, global clock that broadcasts clock signals to all devices in a system so that the entire system operates in a lock-step fashion.

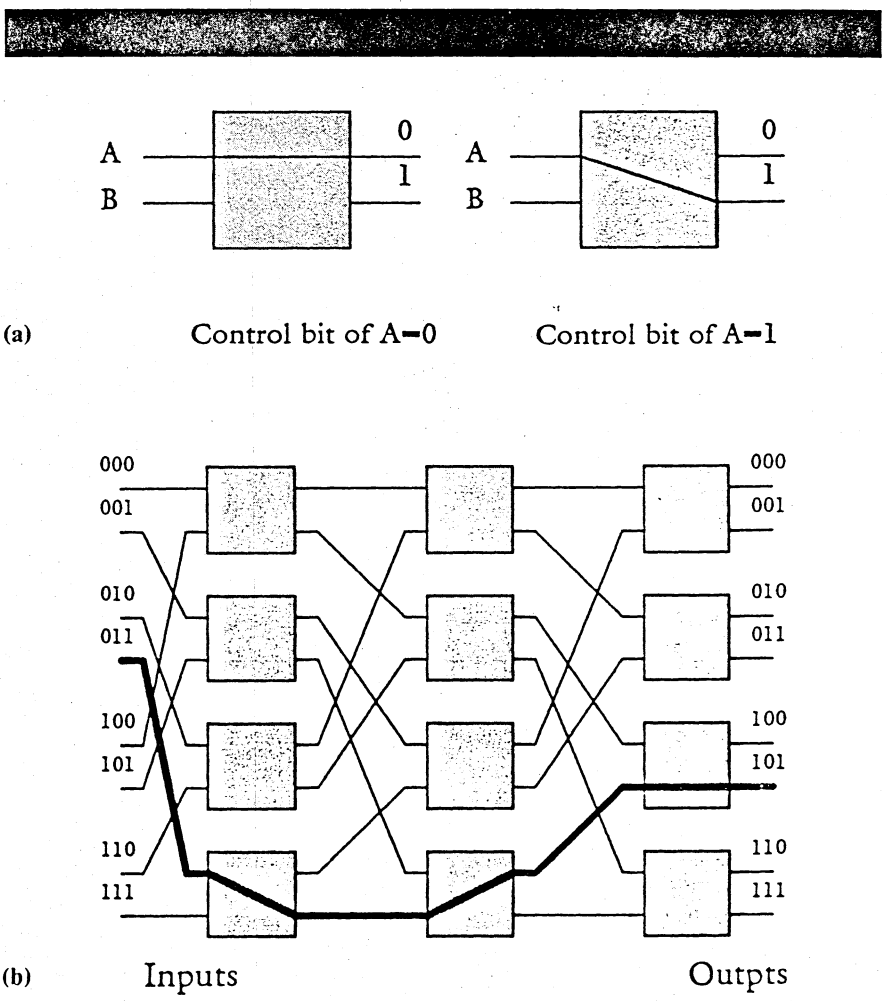


Figure 4. Operation of a 2×2 switch in 4a, and an 8×8 omega network in 4b.

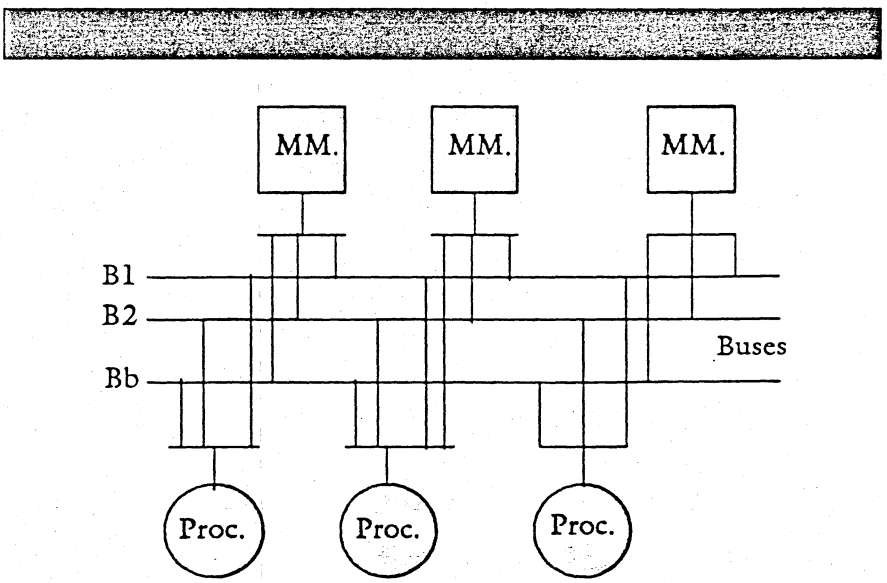


Figure 5. A multiple-bus multiprocessor system.

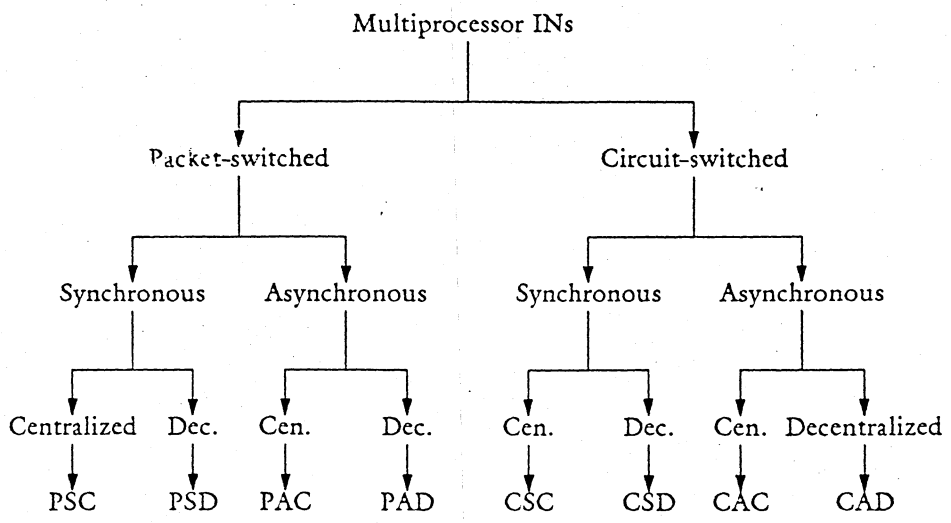


Figure 6. A classification of multiprocessor interconnection networks.

Asynchronous techniques, on the other hand, operate without a global clock. The communications among operational units in the system are performed by means of interlock hand shaking. As a result, they have good expandability and modularity, but are difficult to design.

Switching methodology. There are basically two major switching methodologies: packet switching and circuit switching. In packet switching, a message is broken into small packets transmitted through the network in a "store-and-forward" mode. Thus, a packet experiences a random delay at each switching point, depending on the traffic in the network along its path to the destination.

Conversely, circuit switching actually establishes a physical path between a source and a destination. A time delay is needed when the path is being established. Once the path is completed, it is held for the entire data transmission. In general, circuit switching is much more suitable for long messages, and packet switching is more efficient for short messages.

Control strategy. Control strategy mainly concerns the way control signals direct the dataflow generated in a network. In a centralized control scheme, all the control signals come from a single source. Obviously, the central controller creates a system bottleneck and directly affects the performance and reliability of the entire system. The design of this central controller must be very complex to retain good system performance. These drawbacks can be avoided through the use of distributed control strategies in which a small controller is associated with each component of the system. In multiprocessor applications, control of crossbar networks is usually centralized and control of MINs is usually decentralized. Multiple-bus IN control can be either centralized or decentralized.

Based on the operational characteristics above, INs can be classified into eight different categories for a given topology. The detailed classification scheme is shown in Figure 6. For example, PSC means a packet-switched, synchronous, centrally controlled IN. Together with the topology, these three operational characteristics define an IN. We will examine the performance models of the INs based on this classification scheme.

Basic terminologies for performance evaluation

Before we describe performance analyses of different INs, we need to define several terms. Many performance parameters are applicable for INs. Memory *bandwidth* (*BW*) is the most common performance parameter used in analyzing a synchronous IN in a multiprocessor. It is defined as the mean number of active memory modules in a transfer cycle of the IN. In this case, the term "active" means a processor is successfully performing memory operation (either read or write) in that memory module. *BW* also takes into account the memory access conflicts caused by the random nature of the processors' requests.

Another parameter often used in synchronous analysis, *probability of acceptance* (P_A), is defined as the ratio of expected bandwidth to the expected number of requests generated per cycle.

In asynchronous operation, the *throughput* (*Thr*) of a network is defined as the average number of packets delivered by the network in unit time. In a multiprocessor IN, throughput is the mean number of memory access completions per unit time.

Processor utilization (P_u) is also used as a performance measure and is defined as the expected value of the percentage of time a processor is active. A processor is said to be active when it is doing internal computation without accessing the global memory. Processing power is a simple extension of P_u, which is the sum of processor utilizations over the number of processors.

Other performance parameters can be easily related to the parameters above by applying Little's Law.[2] Moreover, P_u, *BW*, and *Thr* can also be related as

$$P_u = \frac{BW}{N\lambda T}$$

$$P_u = \frac{Thr}{\lambda}$$

where N is the number of processors, T is the time taken for a memory read or write operation, and λ is the memory request rate.

Analytical modeling is a cost effective technique used to study the performance of a computer system. However, any real system is too complex to be modeled exactly.

To make an analytical model tractable, certain approximation assumptions are necessary. Most of the IN analyses assume identical processors and a *uniform reference model*. The URM implies that, when a processor makes a memory request to the global memory, the request will be directed to any one of M memory modules with the same probability $1/M$. That is, the destination address of a memory request is uniformly distributed among M memory modules. This assumption provides us with the symmetric property, significantly simplifying the modeling.

If the memory system is M-way interleaved, this assumption also represents the program behavior reasonably accurately. When the main memory is not interleaved, there is a locality of reference and a favorite memory assumption[3] is more accurate.

The *request rate* of a processor identifies how often a processor accesses global memory. This indirectly reflects the average execution time of an instruction.

In synchronous systems, the request rate can be specified by a probability that a processor generates a memory request at the beginning of a cycle. In asynchronous systems, on the other hand, a memory request could be generated at any instant in time since there is no global clock. However, an exponential thinking time for a processor is commonly assumed, which means that the duration between the completion of a request and generation of the next request to the global memory is an exponentially distributed random variable.

The *request independence assumption* (also called Strecker's approximation[4]) in a synchronous system analysis states that a memory request generated in a cycle is independent of the requests of the previous cycles. In reality, this is not true because a request that was rejected in the previous cycle will be resubmitted in the current cycle. However, as we shall see, this assumption simplifies the analysis to a great extent while keeping the results reasonably accurate.

Performance of crossbar interconnection networks

A crossbar interconnection network is an array of individually operated contact pairs in which there is one pair for each input-output combination, as shown in Figure 3. A crossbar network with N inputs and M outputs is referred to as an $N \times M$ crossbar. As long as there is no memory interference among a set of memory requests generated by the processors (that is, no two or more processors request the same memory module), all connections can be established at the same time. Thus, all memory accesses can proceed simultaneously.

But this capability comes at a high switching cost, which is $(O(NM))$. Although the crossbar network can provide all simultaneous connections, memory bandwidth is much less than its actual capacity. This reduction is due to the memory interference caused by the random nature of the memory requests in a multiprocessor environment. Therefore, the performance analysis of a crossbar network becomes the analysis of memory interference.

The literature[5] contains a number of memory interference models for centralized, synchronous, and circuit-switched crossbar systems. In most of these models, system operations are approximated by stochastic processes as follows: At the beginning of the system cycle, a processor selects a memory module at random and makes a request to access that module with some probability p. If more than one request is made to the same memory module, the memory controller will choose one at random, and the rejected processors will retry in the next cycle. The behavior of the processors is considered independent and statistically identical, as is the behavior of the memory modules.

Bhandarkar[4] studied the memory interference problem in detail in which several discrete Markov chain models were developed. In these models, a memory module is characterized by its cycle time t_c, which consists of an access time t_a, followed by a rewrite time t_w. Processor behavior is modeled as an ordered sequence, consisting of a memory request followed by a certain amount of execution time t_p. The processing time t_p is measured from the time data were obtained from the previous request to the time the next request is issued to the memory.

In real systems, the processor can start execution when the memory is in its rewrite cycle. So, when $t_w = t_p$, the situation would be equivalent to the case where the processor generates a memory request at the beginning of each memory cycle. In this study, an exact model for the case $t_p = t_w$ and with URM was presented. However, the model becomes very unwieldy for a large number of processors and memory modules.

The complexity of the memory interference model is simplified if one assumes that a blocked processor discards the request and generates a new independent request at the start of the next cycle (request independence assumption). For a system with N processors and M memory modules, if a processor generates a request with probability p in a cycle directed to each memory with equal probability (URM), then the memory bandwidth is given by Strecker[4] as

$$BW = M\left(1-\left(1-\frac{p}{M}\right)^N\right) \qquad (1)$$

A simple explanation of this formula is as follows: Since p/M is the probability that a processor requests a particular memory module, $[1-(p/M)]^N$ is the probability that none of the N processors requests the memory module in a particular cycle. Subtracting this term from 1 gives the probability that at least one request to this memory is issued. Multiplying by M yields the expected number of distinct memory modules being requested in a cycle and hence the bandwidth. The maximum percentage of error with this approximation is limited to 8 percent for $M/N > 0.75$.[4] As a result, this simple formula, Equation 1, is widely used for predicting the performance of crossbar networks. The accuracy can be further increased by a "rate adjustment" technique[6] where the input request rate is adjusted upward to take into account the resubmission of rejected requests. Yen[6] provides a comparison of various memory interference models for synchronous crossbars.

The model described above assumes a URM. However, as mentioned previously, the distribution of memory requests in real systems depends on program behavior, and such distributions are not necessarily uniform. Bhuyan[3] has examined this nonuniform reference problem by introducing the concept of favorite memory of a processor. The memory module requested most often by a processor is

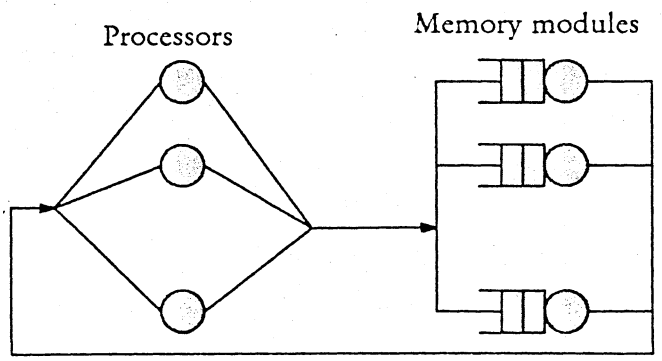


Figure 7. A queueing model for asynchronous crossbar multiprocessors.

called the favorite memory of the processor. Let m represent the probability with which a processor addresses its favorite memory given that the processor generates a request in a cycle. Then, the memory bandwidth for an $N \times N$ crossbar-based multiprocessor is given by

$$BW = N\left[1-(1-pm)\left(1-p\frac{1-m}{N-1}\right)^{N-1}\right] \quad (2)$$

Solutions for favorite memory cases are also provided for $M \leq N$ and $M \geq N$.[7] By substituting $m = 1/M$, the analysis reduces to that of URM (Equation 1).

In the descriptions above, we have considered only circuit-switched and synchronous systems. The analysis of asynchronous circuit-switched systems can be done by assuming a random period of processor thinking time and memory access time. The processors are then modeled by a set of delay servers and memory modules by a set of first-come, first-serve (FCFS) queues, as shown in Figure 7. This figure depicts a well-known closed queueing network in performance evaluation, and efficient algorithms to solve this network exist.[2]

Because the crossbar network is a single-staged network (that is, every input and output is connected by a single switching element), packet switching makes no difference from circuit switching from a performance point of view. Similarly, two control strategies result in the same system behavior. Thus, we need not consider them separately. Table 1 summarizes the different analytical techniques of the crossbar system and their accuracy, workload representations, performance metrics, and computational costs.

Analyses of multistage interconnection networks

As stated previously, the cost of a crossbar network is too high to be practical for building large multiprocessor systems. As an alternative to the crossbar network, multistage interconnection networks (MINs) have assumed importance in recent times. The main advantage of these networks is their cost-effectiveness. They allow a rich subset of one to one and simultaneous mappings of processors to memory modules, while reducing the hardware cost to $O(N\log N)$ in contrast to $O(N^2)$ for crossbar networks.

An $N \times N$ MIN connects N processors to N memories. For N a power of two, it employs $\log_2 N$ stages of 2×2 switches with $N/2$ switches per stage. Each switch has two inputs and two outputs. The connection between an input and an output is established depending on a control bit c provided by the input. When $c = 0$, the input is connected to the upper output, and when $c = 1$, it is connected to the lower output, as shown in Figure 4a.

An omega network[5,8], shown in Figure 4b, is characterized by a perfect shuffle interconnection preceding every stage of switches. The requesting processor generates a tag that is the binary representation of the destination. The connection of a switch at the ith stage is then accomplished by the ith bit of this binary tag counted from the most significant bit.

The connection between input 3 and output 5 (101_2) is shown by a bold line in Figure 4b. This self-routing property of a MIN avoids the need for a central controller, making it very suitable for multiprocessors. Thus, the performance discussions presented in this section will concentrate solely on the decentralized control scheme.

Many significant MINs, such as Banyan, generalized cube, base line, etc.,[8] have been proposed. However, most of these networks are similar except for the interconnection between the adjacent stages.

The switch size in an MIN need not be restricted to 2×2. In fact, the Butterfly parallel processor connects N inputs to N

Table 1. Summary of crossbar analyses.

	Synchronous crossbar		Asynchronous crossbar
Analysis technique	Discrete Marcov chain[4]	Probabilistic with independence assumption[3,4]	Queueing network[2]
Workload representation	Request rate	Probability of request	Think time
Performance parameters	BW or P_A	BW or P_A	P_u or P_W
Accuracy	Exact	Good	Exact
Computational cost	Very high	Low (closed form formula)	Moderate

outputs using 4×4 crossbar switches and $\log_4 N$ stages with $N/4$ switches per stage. A delta network can connect $M = a^n$ inputs to $N = b^n$ outputs through n stages of $a \times b$ crossbar switches.[9] The generalized shuffle network (GSN) is capable of connecting any $M = m_1 * m_2 * \ldots * m_r$ inputs to $N = n_1 * n_2 * \ldots * n_r$ outputs through r stages of switches.[7] The ith stage employs $m_i \times n_i$ crossbar switches and is preceded by a generalized shuffle interconnection that is essentially a superset of the omega and delta interconnections. This is the most generalized version of an MIN that allows different input and output sizes, and all the other networks can be obtained by choosing the m_is and n_is, appropriately. For example, when $m_i = a$, $n_i = b$ for all is, it is a delta network; $m_i = n_i = 2$ for all is gives an omega network; $r = 1$ gives a crossbar; and $M = M * 1$ and $N = 1 * N$ provides a shared-bus connection.

The advantages of MINs were widely recognized by researchers, and a lot of research projects started at universities and industries. Examples of university projects include TRAC (the Texas Reconfigurable Array Computer) at the University of Texas at Austin, Pasm (partitionable single instruction, multiple data [SIMD], multiple instruction, multiple data [MIMD]) at Purdue University, Ultra-Computer at New York University, and Cedar at the University of Illinois at Urbana-Champaign. RP3 is a notable industry project at IBM, and Butterfly is a successfully marketed product by BBN Laboratories.

As these projects were starting, a serious drawback of the MINs surfaced. There is only one path from an input to an output. It was necessary to incorporate some fault-tolerance into these networks so that at least a single fault in a switch or a link could be tolerated. This has given rise to an abundance of research during the past few years devoted to the design and evaluation of fault-tolerant MINs. Adams[10] contains a survey and comparison of such fault-tolerant networks. The evaluation techniques for basic MINs are explained below, but can be extended to fault-tolerant MINs.

Patel[9] suggested a probabilistic approach to analyze the delta network based on URM and the request independence assumption. Assume a delta network of size $a^n \times b^n$ constructed from $a \times b$ crossbar modules. Each stage of the delta network is controlled by a distinct destination digit (in base b) for setting of individual $a \times b$ switches. Since the destinations are independent and uniformly distributed, the requests at any $a \times b$ modules are independent and uniformly distributed over b different destinations. In addition, the switches at a particular stage behave similarly. Therefore, Equation 1 can be applied to any switching element in the delta network.

The expected number of requests that pass to the b outputs is obtained by setting $N = a$ and $M = b$ in Equation 1. Dividing this number by b gives us the probability of request on any of the b output lines of an $a \times b$ switch as a function of its input probability. Since the output of a stage is the input of the next stage, one can recursively evaluate the output probability of any stage starting at stage 1. If p_i is the probability that there is a request at the output of a switch at stage i, then

$$p_i = 1-(1-\frac{P_{i-1}}{b})^a \qquad (3)$$

for $1 \le i \le n$. In particular, the output probability of the final stage determines the bandwidth of a delta network, that is, $BW = p_n b^n$. This analytical technique has been widely used to evaluate various MINs.

Bhuyan[3] extended the analysis to favorite-memory cases. For $N \times N$ networks, with $N = a^n$, the processors are defined to be connected to their favorite memories when all the switches are straight connected, that is, input i of a switch is connected to the output i of the switch. In an omega network memory, MM_j becomes the favorite memory of processor P_j. Let q_{i-1} be the probability that there is a favorite request to the input at stage i. In an $N \times N$ ($N = a^n$) delta network,

$$p_i = 1-(1-p_{i-1}q_{i-1})\,(1-p_{i-1}\frac{1-q_{i-1}}{a-1}) \qquad (4)$$

About six other equations are needed to determine q_{i-1} at stage i[3] and, finally, $BW = Np_n$. The analyses above[3,9] are valid for synchronous packet-switched MINs provided that (a) packets are generated only at the beginning of the network cycle and (b) switches do not have buffers (are unbuffered) so that packets are randomly chosen in case of conflicts and unsuccessful packets are lost.

The above analyses presented a recurrence relation for the performance of unbuffered networks, but not a closed-form solution. Kruskal and Snir[11] obtained an asymptotic expression for the output request probability of a stage for an unbuffered delta network. Let p_m denote the probability that there is a packet on any particular input at the mth stage of a square MIN composed of $k \times k$ switches. Through some algebraic manipulations, Kruskal and Snir[11] approximated the asymptotic formula for p_m as

$$p_m = \frac{2k}{(k-1)m+\frac{2k}{p}} \qquad (5)$$

where p is the probability of request generation by a processor. From this expression, one can see that the probability that a message is not deleted is inversely proportional to the number of stages in the network. The solution for an unbuffered network by Kruskal and Snir has been shown to be a strict upper bound in the throughput of a delta network. For buffered networks, Kruskal and Snir assume an infinite buffer associated with each output of a switching element. In each cycle, a random number of packets join an output queue without any packet being lost. This random number has a Bernoulli distribution, since all incoming packets from the inputs of the switch have an equal probability (for URM) of going to that output. The average transit time through the network can be derived by means of the $M/G/1$ queueing formula.[11]

Dias and Jump[12] have studied buffered delta networks by means of petri nets, which were introduced first as a useful graphical tool for the precise description of the system operations and as a modeling technique that permits easy evaluation of the performance of small systems.

These graph models have recently become very popular for the representation of distributed computing systems because of their ability to clearly describe concurrency, conflicts, and synchronization of tasks. However, the complexity of petri nets increases exponentially with the increase in system size. With this modeling technique, 14 distinguishable states of a (2×2) switch exist with a single buffer between stages.[12] The state transition tables and the probabilities in each state are derived. The steady state throughput and turnaround time (network delay) are obtained by iterating through use of the transition tables and probability equations. The results of analysis and simulation indicate that buffering produces a considerable improvement in the performance of these networks.

All the analyses above pertain to synchronous circuit-switched or packet-switched environments. For large MINs,

Table 2. Summary of MINs analyses.

	Synchronous MIN without buffer	Synchronous MIN with infinite buffers	Synchronous MIN with finite buffers	Asynchronous MIN with finite buffers
Analysis technique	Probabilistic with independence assumption[3,9]	$M/G/1$ queue with infinite buffers[11]	Petri net[12]	Multiple chain MVA[5]
Workload representation	Request rate	Request rate	Processor think time	Processor think time
Performance parameters	BW or P_A	Queueing delay or transit time	Throughput or turn-around time	Throughput or response time
Accuracy	Good	Fair	Good	Good
Computation cost	Low for recurrence solutions	Closed-form formula	High	Moderate

controlling the network operation from a central global clock is difficult. Hence, asynchronous designs should be considered for large multiprocessor systems.

The second disadvantage with the above analyses is that they do not incorporate the waiting time of a processor for completion of the memory access, but assume continuous Poisson arrival at the input side.

The third disadvantage is the assumption of uniform or favorite memory access. Memory reference patterns are highly program dependent and could be arbitrary.

To overcome these drawbacks, we have recently developed[5] a closed queueing network model and mean value analysis (MVA)[5] for the MINs under asynchronous packet-switched operation. Modeling asynchronous circuit-switched MINs seems very difficult because of the simultaneous possession of switches and links by unsuccessful requests. Table 2 lists the different analyses of MINs described in this section.

Performance analyses of multiple-bus systems

Most commercial systems containing more than one processor employ a single shared bus as shown in Figure 2. This interconnection scheme is well known as being inexpensive and easy to implement. But when the system size is large, a single bus becomes a severe system bottleneck.

A natural extension is to employ several buses instead of a single one to increase the bandwidth and fault tolerance at moderate cost. Recently, the multiple-bus interconnection scheme has drawn considerable attention from many computer scientists and engineers. In an $N \times M \times B$ multiple-bus multiprocessor system, all the N processors and M memory modules are connected to all the B buses.

Unlike a crossbar or multistage network, the multiple-bus configuration offers high reliability, availability, and easy incremental system growth. Higher reliability is obvious because, in case of a bus failure, $(B-1)$ distinct paths still exist between a processor and a memory. However, when the number of buses is less than the number of memory modules or the number of processors, bus contention can arise. As a result, the performance analysis of a multiple-bus system involves modeling the effects of bus conflicts and memory interference.

Many researchers have studied the performance of synchronous, circuit-switched, and centrally controlled multiple-bus multiprocessor systems through analysis and simulation.[5,13] The memory bandwidth of the system increases with an increase in the number of buses. But, for all practical purposes, a few buses might be sufficient.

In a synchronous system, all the events occur at the beginning of a system cycle. Therefore, the system can be modeled by means of a discrete Markov process. Bhuyan has developed a combinatorial and probabilistic approach to derive an equation for the BW of such multiple-bus multiprocessor systems.[5,13] His analysis is based on the URM and request independence assumptions.

With these assumptions in mind for a given set of memory requests, knowing the number of ways in which the requests are distributed among the memory modules is easy. In addition, one can determine the number of ways by which the given requests are addressed to M memory modules such that x memory modules are requested and $M-x$ of them are idle.

For each value of x, Bhuyan defines a state. The BW can be computed by multiplying the probability of being in a state with the number of busy memory modules in that state. If the number of buses in a system is less than the number of memory modules, the number of busy memory modules in a cycle would be upper bounded by the number of buses. The bandwidth for an $N \times M \times B$ system is given by

$$BW = M\left\{1-\left(1-\frac{p}{M}\right)^N\right\} - \sum_{y=B+1}^{N}\binom{N}{y}p^y(1-p)^{N-y} \times \sum_{x=B+1}^{t_y}\frac{(x-B)x!\binom{M}{x}S(y,x)}{M^y} \quad (6)$$

where, $x!$ is the factorial x, $t_y = min(y,M)$, p is the probability of request of a processor, $\binom{N}{y}$ is the binomial coefficient, and $S(y,x)$ is the Stirling number of the second kind defined as

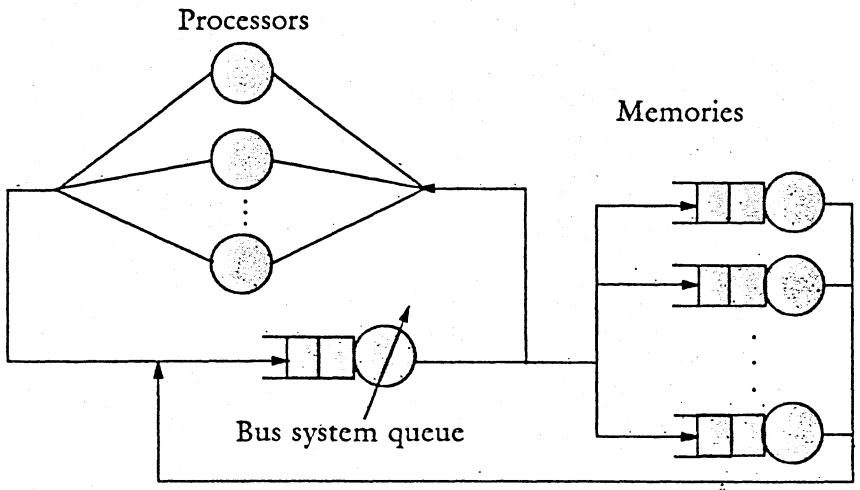


Figure 8. A queueing network model for asynchronous multiple-bus systems.

$$x!\, S(y,x) = \sum_{i=0}^{x} (-1)^i \binom{x}{i} (x-i)^y$$

The numerical results obtained from this equation are quite close to simulation results. Mudge[13] contains a detailed review and comparison of various analyses on synchronous circuit-switched multiple-bus systems.

In asynchronous systems, memory request generation and memory access completion can occur at any point in time since there is no synchronization by a clock. Therefore, the system can be modeled with a continuous stochastic process. Each shared resource, such as a bus or a memory module, is considered a queueing service center.

Because of circuit switching, the processor that issues a memory request holds a system bus while accessing the main memory. Thus, two system resources, bus and memory, are simultaneously held by a memory operation. This simultaneous resource possession phenomenon makes the analysis nontrivial.

One study[14] uses the *flow equivalence* technique to approximately solve the queueing network. The bus and memory subsystem, called aggregate, is replaced by a single flow equivalent service center (FESC).[2] The model has been compared with simulation results, and the agreement is quite good over a wide range of bus load.[14]

All the studies above consider only centrally controlled, circuit-switched multiple-bus systems. In circuit switching, a device will occupy the bus for the entire duration of data communication once the device is granted use of a bus.

For instance, a processor in a read operation will occupy the bus during the time it is sending a request, performing a memory operation, and receiving the requested data. The result will be the waste of a significant fraction of bus bandwidth because of a mismatch between the speeds of the processing unit, the bus, and the memory unit.

All these facts serve to demonstrate the attractiveness of the packet-switching approach. Encore's Multimax is an example of a multiprocessor employing a packet-switched shared bus between processors and memories.

Yang studied[15] packet-switched multiple-bus systems where analytical models were developed for both synchronous and asynchronous timings. In the synchronous case, both centralized and decentralized controlled schemes are treated equally, since all the events in the system occur at the beginning of a cycle regardless of the control strategies used. A discrete probabilistic approach is applied to analyze such systems.

The model has been based on a decomposition technique that considers simple analysis of a set of single-server queues. The consequence of the analysis is an equation consisting of one unknown variable, P_u, processor utilization; it can be solved by using a standard numerical method.

For an asynchronous case, the queueing network is shown in Figure 8. Processors in the system are modeled as delay servers, and memory modules are modeled as FCFS servers. The bus system is modeled as an FESC[2] representing B buses with a single (centralized) queue.

The routing of a packet in the network can be described as follows: A request packet generated by a processor is first put in the central server queue, waiting for an available bus. After it gains access to a bus, the packet joins one of the M memory queues. The memory module that finishes the service of a request again puts the response packet in the central server queue. From there, the response packet gets back to the requesting processor through a bus. To this point, the packet finishes one rotation through the network, and the processor resumes its background activity. The model can be solved using any standard product-form algorithm, considering the bus system queue as a load-dependent server.[2]

In the case of a decentralized control, the actual implementation can be either token bus or daisy chain bus. Due to the lack of central controller, the analysis of decentralized packet-switching multiple-bus systems is very complicated. The FCFS service discipline is not valid for bus system queue in this case. The service discipline depends on the position of tokens with respect to the positions of requesting devices.

The exact solution for such a system seems infeasible, but one method for approximating the behavior of the system is to use hierarchical modeling tech-

Table 3. Summary of multiple-bus analyses.

	Circuit-switched synchronous	Circuit-switched asynchronous	Packet-switched synchronous	Packet-switched asynchronous
Analysis technique	Probabilistic with independence assumption[5,13]	Queue network with infinite buffers[14]	Probabilistic queueing analysis[15]	Queueing network with FESC[15]
Workload representation	Request rate	Processor think time	Request rate	Processor think time
Performance parameters	BW or P_A	Throughput or P_u	P_u	Throughput or P_u
Accuracy	Fair	Good	Good	Fair
Computation cost	Low	Moderate	Low, but iteration	Moderate

Table 4. Summary of hardware features of the three INs.

	Crossbar	MINs	Multiple-bus
No. switches or connections	$N*M$	$N\log N$	$B*(N+M)$
Load of buses	N	1	B
No. of wires	M	N	B
Arbiter	M 1-of-N arbiters	$N\log N$ 1-of-2 arbiters	1 B-of-M and M 1-of-N arbiters
Fault-tolerant and expansion	Fair	Poor, but fair with additional hardware	Good

niques.[15] As in the previous case, the bus system is represented by FESC (Figure 8) with the service rate obtained by shorting out the processors and the memory modules. The model is then solved by using the mean value analysis algorithm.[2]

Numerical results obtained from the models have shown that packet switching reduces the communication bottleneck of shared bus systems.[15] Table 3 summarizes analyses of different categories of multiple-bus systems.

Comparison and discussions

As indicated in the previous sections, the three types of interconnection networks possess different hardware features and different system performances. In this section, we will look into these differences quantitatively. In particular, we will compare the hardware cost and system performance of the three interconnection networks.

Table 4 lists selective hardware features of the three networks. As we already know, the number of switching elements used in an $N \times M$ crossbar is $N \times M$ in contrast to $N\log N$ of MINs. The number of connections necessary in an $N \times M \times B$ multiple-bus system is proportional to $B(N + M)$. Since each bus in the multiple-bus system needs to drive $N + M$ modules, the bus load is proportional to $N + M$, while the bus load of an MIN is one due to the one-to-one connection.

All three of these networks require certain types of arbiters to resolve the request conflicts. In a crossbar network, M N-users 1-server arbiters are necessary, each of which selects one of up to N outstanding requests for a memory during a memory cycle. The MINs, on the other hand, require two 2-user 1-server arbiters for each switching element for $(N/2)\log_2 N$ switches.

In a multiple-bus system, an M-users B-servers arbiter is needed to assign the B buses to the outstanding requests. Once a bus is granted to a memory, only one of the processors that requests the memory can proceed while the others, if any, are delayed. This choice is implemented by an N-users 1-server arbiter. Thus, a multiple-bus system requires $M + 1$ arbiters, one arbiter of the M-users B-servers type and M arbiters of the N-users 1-server type.

Expandability and reliability are two other very important hardware features. In this context, a multiple-bus system shows its advantages over the other two because of its reconfigurability and multiple-data paths between every processor and memory. It can still operate in a

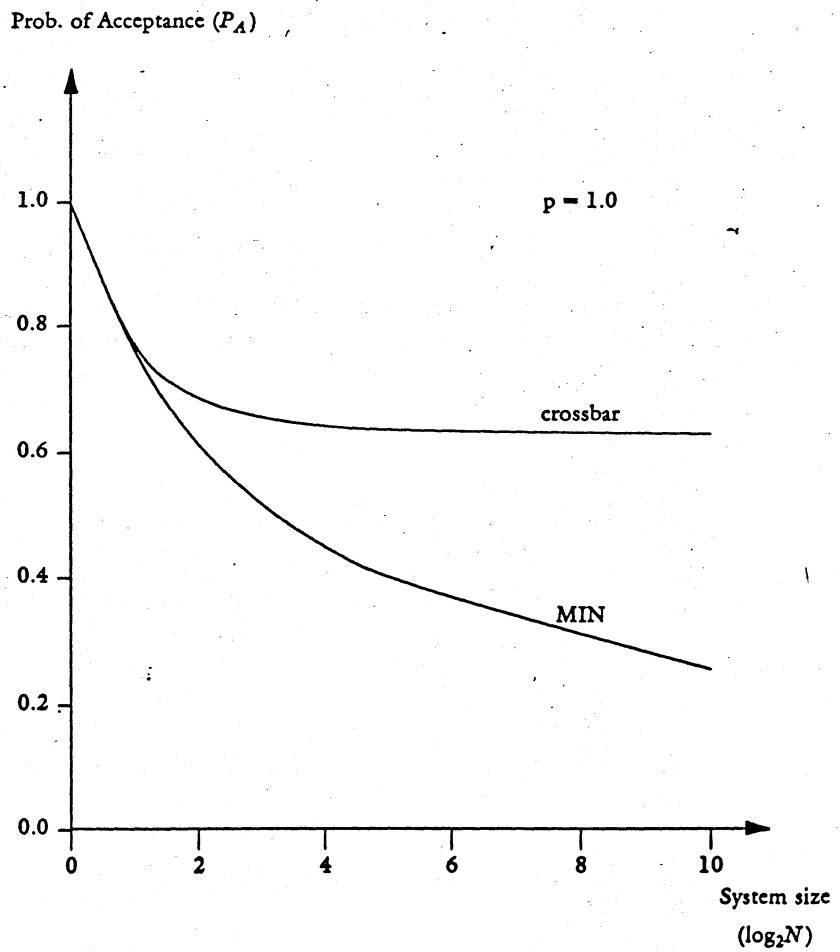


Figure 9. Probability of acceptance as a function of system size for synchronous circuit-switched systems.

degraded mode after the failure of a subset of the buses.

A lot of research has gone into the reconfigurable and fault-tolerant MINs.[10] The fault-tolerance of a MIN can be realized by adding additional hardware such as extra stages or duplicated data paths. It is easier to reconfigure a crossbar than an MIN. In case of a fault, a particular row or a column in Figure 3 can be removed and the network can operate in a degraded mode.

Next, we'll consider the performance of the three interconnection networks based on the analytical models described in the previous sections. Figure 9 shows the probability of a memory request being accepted, $P_A = BW/p \cdot N$, as a function of system size for synchronous circuit-switched systems with $p = 1.0$.

Two curves, one for crossbar and one for MIN, are plotted according to Equation 1 and Equation 3, respectively. The difference between the two curves increases as the system size grows. The probability of acceptance in the crossbar system remains constant when the system size becomes very large. However, in the case of MIN, P_A keeps decreasing as the system size increases.

Figure 10 shows the memory bandwidth of 16×16 synchronous circuit-switched multiprocessor systems. The horizontal axis represents a processor's probability of request. As we can see, the single bus performs worse since it gets saturated very quickly and the BW can never exceed 1. The BW increases as the number of buses increases, as shown in the figure.

Figure 11 shows the comparison of synchronous packet-switched networks. In this figure, processor utilizations are plotted against the probability of request for a 16×16 multiprocessor. Processor cycle times, bus transfer delay, as well as the total ideal packet transfer time between an input and an output of an MIN are assumed to be fixed at the system cycle time. The memory access time is assumed to take four system cycles (similar to Encore's Multimax) for all three systems. During a memory access, the processor that issued the memory request remains idle until the memory access is finished.

The memory request rate of a processor as seen by an interconnection network is adjusted[6] in plotting the processor utilization in Figure 11. The performance difference between various networks is not as pronounced as in Figure 10. This is due to the packet-switched operation of the INs. Additionally, a multiple-bus-based multiprocessor system with only four buses can achieve almost the same performance as that of the crossbar system while reducing the hardware cost significantly.

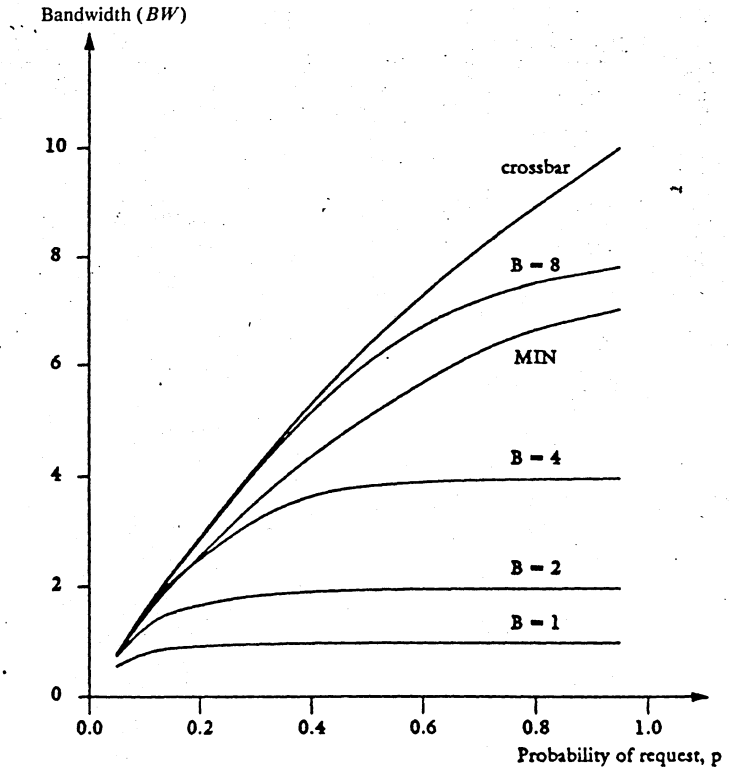


Figure 10. Memory bandwidth as a function of probability of request for 16 × 16 synchronous circuit-switched systems.

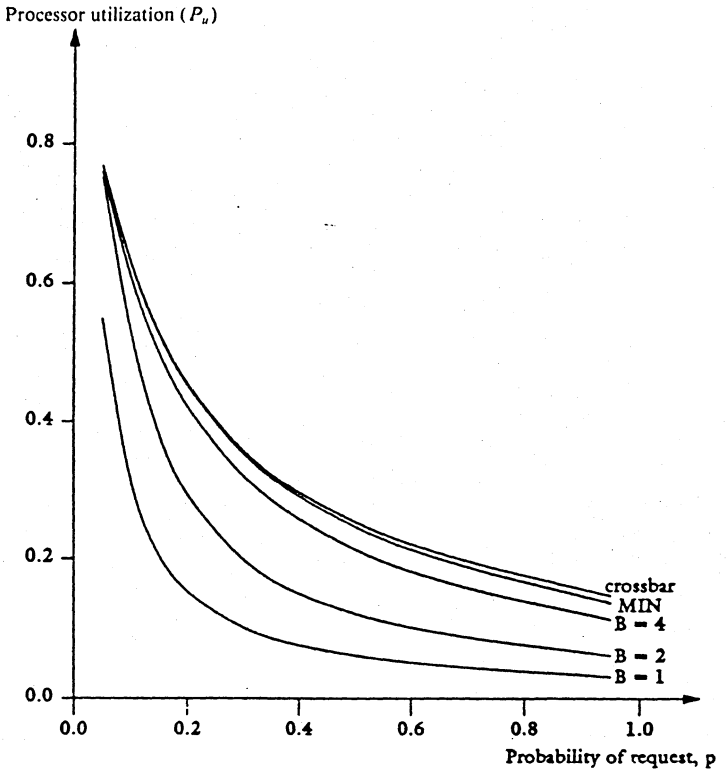


Figure 11. Processor utilization as a function of probability of request for 16 × 16 synchronous packet-switched systems.

All the comparisons above apply to synchronous systems based on system cycle. We have intentionally avoided comparing asynchronous systems because their performance is so dependent on the input parameters that choosing the wrong parameters might give rise to the wrong conclusions. However, the analytical techniques that can be applied to evaluate those systems are given in this article. The information provided here is useful for predicting the approximate performance of an IN structure before its design and implementation. Further references and a more detailed survey on the performance evaluation of multiprocessor INs can be found in Bhuyan.[5]

It seems that enough research has already been done in evaluating INs in isolation. We strongly feel that more work is needed at the system level that includes the IN as a major component. For example, evaluation of multiprocessor systems with prefetching, bulk data read or write, and solving cache coherence with INs shows promise for future research.

Similarly, task (application) level modeling on multiprocessor architectures might produce some good insight into the trade-offs between computation versus communication, low versus large granularity, static versus dynamic scheduling, etc.

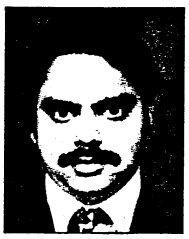
Finally, some actual measurements (traces) should be obtained on real multiprocessors and applied to the analytical models as their input parameters. □

Acknowledgments

Bhuyan's research was supported by National Science Foundation grant No. MIP-8807761 and by a grant from the Louisiana Board of Regents. Agrawal's research was partly supported by US Army Research Office contract No. DAAG 29-85-K-0236.

References

1. D.A. Reed and D.C. Grunwald, "The Performance of Multicomputer Interconnection Networks," *Computer,* Vol. 20, No. 6, June 1987, pp. 63-73.
2. E.D. Lazowska et al., *Quantitative System Performance—Computer System Analysis Using Queueing Network Models,* Prentice Hall, Englewood Cliffs, N.J., 1984.
3. L.N. Bhuyan, "An Analysis of Processor-Memory Interconnection Networks," *IEEE Trans. Computers,* Vol. C-34, No. 3, Mar. 1985, pp. 279-283.
4. D.P. Bhandarkar, "Analysis of Memory Interference in Multiprocessors," *IEEE Trans. Computers,* Vol. C-24, Sept. 1975, pp. 897-908.
5. L.N. Bhuyan, "Performance Evaluation of Multiprocessor Interconnection Networks," *Tutorial Note,* ACM SIGMetrics Conf., May 1988.
6. D.W. Yen, J.H. Patel, and E.S. Davidson, "Memory Interference in Synchronous Multiprocessor Systems," *IEEE Trans. Computers,* Vol. C-31, Nov. 1982, pp. 1,116-1,121.
7. L.N. Bhuyan and D.P. Agrawal, "Design and Performance of Generalized Interconnection Networks," *IEEE Trans. Computers,* Vol. C-32, Dec. 1983, pp. 1,081-1,090.
8. Tse-Yun Feng, "A Survey of Interconnection Networks," *Computer,* Vol. 14, No. 12, Dec. 1981, pp. 12-27.
9. J.H. Patel, "Performance of Processor-Memory Interconnections for Multiprocessors," *IEEE Trans. Computers,* Vol. C-30, Oct. 1981, pp. 771-780.
10. G.B. Adams III, D.P. Agrawal, and H.J. Siegel, "A Survey and Comparison of Fault-Tolerant Multistage Interconnection Networks," *Computer,* Vol. 20, No. 6, June 1987, pp. 14-27.
11. C.P. Kruskal and M. Snir, "The Performance of Multistage Interconnection Networks for Multiprocessors," *IEEE Trans. Computers,* Vol. C-32, Dec. 1983, pp. 1,091-1,098.
12. D.M. Dias and J.R. Jump, "Analysis and Simulation of Buffered Delta Networks," *IEEE Trans. Computers,* Vol. C-30, Apr. 1981, pp. 273-282.
13. T.N. Mudge et al., "Analysis of Multiple-Bus Interconnection Networks," *J. Parallel and Distributed Computing,* Vol. 3, No. 3, Sept. 1986, pp. 328-343.
14. D. Towsley, "Approximate Models of Multiple Bus Multiprocessor Systems," *IEEE Trans. Computers,* Vol. C-35, Mar. 1986, pp. 220-228.
15. Q. Yang, L.N. Bhuyan, and R. Pavaskar, "Performance Analysis of Packet-Switched Multiple-Bus Multiprocessor Systems," *Proc. 8th Real-Time System Symp.,* Dec. 1987, CS Press, Los Alamitos, Calif., pp. 170-178.



Laxmi N. Bhuyan is an associate professor at the Center for Advanced Computer Studies at the University of Southwestern Louisiana in Lafayette. His research interests include parallel and distributed computer architecture, performance and reliability evaluation, and local area networks.

Bhuyan received BS and MS degrees in electrical engineering from the Regional Engineering College, Rourkela under Sambalpur University in India. He received a PhD in computer engineering from Wayne State University in Detroit in 1982. Bhuyan is a senior member of the IEEE, a distinguished visitor of the IEEE Computer Society, and served as guest editor of the *Computer* issue on interconnection networks in June 1987.

Qing Yang is an assistant professor in the Department of Electrical Engineering at the University of Rhode Island. His research interests include parallel and distributed computer systems, design of digital systems, performance evaluation, and local area networks.

Yang received a BSc degree in computer science from Huazhong University of Science and Technology in China in 1982 and an MASc in electrical engineering from the University of Toronto in 1985. He received a PhD in computer engineering from the Center for Advanced Computer Studies, University of Southwestern Louisiana. Yang is a member of the IEEE Computer Society.

Dharma P. Agrawal is a professor of electrical and computer engineering at North Carolina State University in Raleigh. He is the group leader for the B_Hive multicomputer project at NCSU. His research interests include both software and hardware aspects of parallel and distributed processing, computer architecture, and fault tolerant computing.

Agrawal received the BE degree from Ravishankar University in India, the ME degree from the University of Roorkee in India, and the DScTech degree in 1975 from the Federal Institute of Technology in Switzerland. He is the author of the tutorial text *Advanced Computer Architecture* published by the IEEE Computer Society; a member of the *Computer* Editorial Board; an editor of the *Journal of Parallel and Distributed Computing* and the *International Journal on High-Speed Computing*; a fellow of the IEEE; a member of the IEEE Computer Society; and a recipient of the society's Certificate of Appreciation.

Readers may write to Bhuyan at the Center for Advanced Computer Studies, University of Southwestern Louisiana, Lafayette, LA 70504-4330.