Performance Analysis and Scheduling of Stochastic Fork-Join Jobs in a Multicomputer System

Anurag Kumar, *Senior Member, IEEE,* and Rajeev Shorey

Abstract—**We model a parallel processing system comprising several homogeneous computers interconnected by a communication network. Jobs arriving to this system have a linear fork-join structure. Each fork of the job gives rise to a *random* number of tasks that can be processed independently on any of the computers. Since exact analysis of fork-join models is known to be intractable, we resort to obtaining analytical bounds to the mean job response time of the fork-join job. For jobs with a single fork-join and, probabilistic allocation of tasks of the job to the N processors, we obtain upper and lower bounds to the mean job response time. Upper bounds are obtained using the concept of associated random variables and are found to be a good approximation to the mean job response time. A simple lower bound is obtained by neglecting queueing delays. We also find two lower bounds that include queueing delays. For multiple fork-join jobs, we study an approximation based on associated random variables. Finally, two versions of the Join-the-Shortest-Queue (JSQ) allocation policy (i.e., JSQ by batch and JSQ by task) are studied and compared, via simulations and diffusion limits.**

Index Terms—**Fork-join parallelism, lower/upper bounds, performance evaluation, queueing models, stochastic scheduling, task allocation policies.**

I. Introduction

THE areas of parallel processing and distributed computing systems have been the focus of a tremendous amount of research in the last decade. The technological limitations on the speed of uniprocessor computer systems have led to the emergence of multiprocessor systems that consist of several, loosely or tightly, interconnected processors. The jobs to be processed are in some way apportioned among the processors, and various techniques are used to coordinate the processing of the various pieces of the jobs.

With the advent of multiprocessors and programming languages that support parallel programming, there is an increasing interest in the performance analysis of parallel programs. In this paper, we study the performance of a particular type of parallel program, a *stochastic fork-join job*, on a multiprocessor system consisting of homogeneous processors interconnected by a communication network.

A stream of jobs having a *linear fork-join* structure arrives to the various nodes of the computing system. Each such job consists of a series of forks and joins, such as those that might be created by "parbegin" and "parend" constructs in parallel programming languages. Each fork gives rise to a random number of tasks that can be processed independently on any of the computers.The job terminates after a random number of fork-joins (Fig. 1). Our objective is to study the response times of such jobs in a multicomputer system.

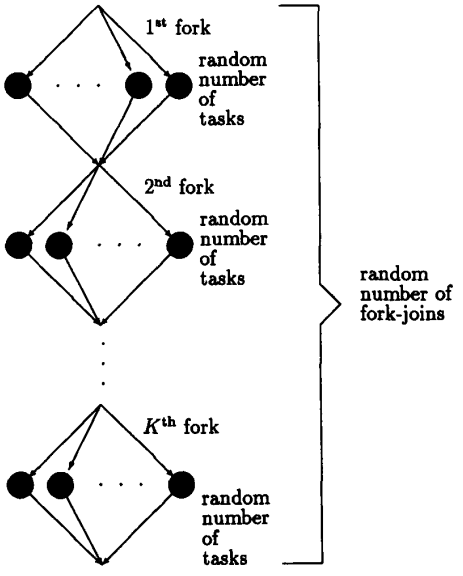


Fig. 1. A job with multiple fork-joins.

In such a system jobs may arrive to a central host computer which allocates tasks to other computers, and is responsible for delivering the final results to the users. This would yield the model in Fig. 2.

Alternatively, users may submit jobs to any of the computers; the computer at which a job "originates" is responsible for allocating tasks to computers (including itself), and for delivering results to the users. This would yield the model in Fig. 3. In this paper we do not model communication delays, the overheads involved in task allocation, and assembly of results. Under this assumption, it is clear that the two job arrival and task allocation scenarios are equivalent. Further, if we assume that in the model of Fig. 3 the job arrival processes are independent Poisson processes, then the model is equivalent to that in Fig. 2 with Poisson job arrivals and $\lambda = \sum_{i=1}^{N} \lambda_i$.

With these assumptions, we obtain the model shown in Fig. 2. This is the *fork-join queueing* model that has two variations from the version studied in the literature; namely:

Manuscript received March 21, 1991; revised April 15, 1992.

The authors are with the Department of Electrical Communication Engineering, Indian Institute of Science, Bangalore 560012, India.

IEEE Log Number 9213473.

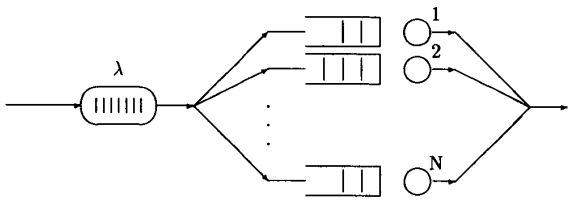


Fig. 2. Multicomputer model: all jobs arrive to a central scheduler.

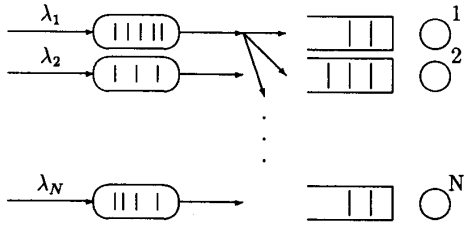


Fig. 3. Multicomputer model: each processor has its own job arrival stream.

1) The number of tasks in each fork is random.
2) There is an allocation policy for tasks created by a fork.
3) A job completes after a random number of forks.

Observe that, except in certain special cases (e.g., Poisson batch size, probabilistic task allocation, and a single fork per job), this model is not equivalent to the fork-join model studied in the literature.

Fork-join queueing models similar to that described above have been studied in the literature in the context of Manufacturing Systems and Parallel Processing Systems [6]. The main difference between the fork-join model we have described and the "standard" one studied in the literature is that in the latter model the task structure of the arriving jobs is *deterministic* (i.e., the number of tasks in a fork is a constant) and the tasks map exactly on to the processing elements, one task to each element. For example, in the most commonly studied model, each job brings N tasks, there are N processors, and one arriving task is scheduled on each processor. It is clear that such a model is appropriate for a manufacturing system, or a dedicated computing system, that repeatedly processes different instances of the same job, but is entirely inappropriate for a general purpose parallel processing system.

Even in these simpler models, the *synchronizations* induced by the forks and joins destroy all nice properties like insensitivity or product form [33], so that the analysis becomes computationally hard. Exact analysis is only possible for very simple system models and computation graphs. Exact solutions have been provided when there are two processors ([10], [2]). Approximate solutions and bounds have been provided for arbitrary values of processors (N) ([5], [20]). Baccelli [5] has considered a fork-join queue consisting of $N \geq 2$ heterogeneous servers, with general arrival and service processes. An *upper bound* is obtained by considering N mutually independent $GI/GI/1$ parallel queueing systems, and a *lower bound* is obtained by considering N $D/GI/1$ parallel queueing systems.

Conditions for stability have been presented for arbitrary values of N ([3], [26]). Finally, models have been developed for programs exhibiting parallel fork-join structures that are executed on multiple processors serving a single queue ([19], [14], [30]). Note that Nelson ([19]) considers a job structure that is not deterministic; i.e., the number of tasks in a job is a random variable. However, the model analysed is that of a *centralized parallel processing system* (and, hence does not yield a fork-join queue) where the N servers are all fed by a single queue, whereas what we model and analyze is a *distributed parallel processing system* with a queue associated with each of the N servers.

In our model where jobs have a fork-join structure, since the number of arriving tasks is possibly different from the number of processors, a task allocation policy needs to be specified. When a job (or, equivalently a task batch) arrives, the policy allocates a subbatch of tasks to each processor. What makes our model more complex to analyze than those studied previously ([20], [5]) is the fact that the subbatches are *dependent*. The dependency between the components of a batch causes the subbatch service times to be dependent. It is worth noting that if we consider *Poisson batch size*, and if we *multinomially partition* the arriving batch over the N processors, then the subbatches are independent. However, in general, there will be a dependency between the subbatches of a job.

The paper is organized as follows. In Section II, we introduce a general model of a parallel processing system. The multicomputer system with N homogeneous processing elements is modeled as a queueing system with N servers. Each job arriving to this queueing system is a random batch of tasks with precedence constraints. Stability conditions are obtained for the general model.

For the remainder of the paper we consider only fork-join queueing models with the stochastic fork-join job structure described above.

Exact analysis of fork-join queueing models is well known to be intractable, hence, as usual, we resort to obtaining analytical bounds. In Sections III and IV we study the situation where there is only one fork-join in each job.

In Section III, we find upper bounds to the mean response time of a job with the fork-join structure. We consider an allocation policy in which the arriving batch is multinomially partitioned over the N processors, each task going to the ith processor with probability p_i. We first obtain an upper bound based on the l_k-norm, and convexity. Another upper bound is obtained using the concept of associated random variables. By comparison with simulations, we find that the first upper bound is quite loose, whereas the upper bound based on associated random variables can also serve as a reasonably good approximation.

Section IV describes various lower bounds to the mean job response time in a fork-join system. One lower bound to the mean job response time is the mean delay in any of the N component queues of the fork-join queueing system. A lower bound that ignores queueing delays is discussed. Finally, we find a lower bound which is a function of both ρ and N.

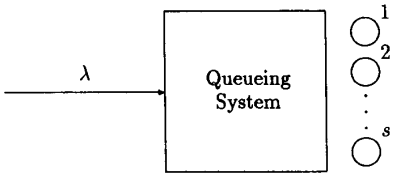


Fig. 4. The general model of a multicomputer.

In Section V, we study a simple approximation for the fork-join queueing model where jobs have a multiple fork-join structure. This approximation is based on associated random variables.

In Section VI, we compare two JSQ (Join-the-Shortest-Queue) allocation policies. We compare JSQ by batch to JSQ by task and study the effect of increasing the load on the mean response time of a fork-join job in these two systems. Our results are based on simulation and diffusion limit analysis.

Finally, in Section VII, we discuss the conclusion and scope for further extensions.

II. The General Queueing Model of a Multicomputer System

We consider a multicomputer system with N identical (homogeneous) processing elements, modeled as a queueing system with N servers. At this point, we do not specify how the tasks are queued up and are allocated to servers. Thus, in Fig. 4, we represent the queueing system as a black box, that could comprise, for example, either a single queue with all tasks joining the queue, or a queue per server with some policy for allocating tasks to queues. Jobs arrive to the system in a continuous random stream, *each job is a random batch of tasks* with precedence constraints. The precedence relation between the tasks in a job constitutes a partial order that can be represented by a Directed Acyclic Graph (DAG) [31].

Tasks of a job are assigned in their entirety to the processors (servers), and at each processor tasks are served to completion, i.e., we do not allow preemption of tasks. The processing of a job is complete when all its constituent tasks have been processed, in accordance with the precedence requirements. The time duration between the arrival of a job and the completion of its processing is the job response time (or sojourn time).

We adopt a convention whereby sets are denoted by Greek letters in *upper case*, say Γ, Θ, etc. Consider a set of tasks $\Theta = \{T_1, T_2, \cdots, \}$. These tasks constitute a job. Let $\prec$ be a partial binary relation on Θ, with $T_i \prec T_j$ meaning that task i in a job precedes task j. Define $\Gamma \stackrel{\text{def}}{=} \{(i,j) : T_i \prec T_j\} \subseteq \Theta \times \Theta$. We require that $\prec$ be transitive and antisymmetric. Thus, Γ can be represented by a directed acyclic graph [31].

We begin in a general setting by assuming that the job arrival epochs form a renewal point process, the jobs comprise batches of tasks with independent and identically distributed (i.i.d) batch sizes, and the task service times are i.i.d random variables (from job to job and within a job). So far we have left the queueing and service discipline unspecified.

Let B denote the task service time random variable. Let C denote the random number of tasks in a batch. Let λ be the arrival rate of jobs.

We denote this class of models by $GI^{\mathcal{P}}/GI/N$, where $\mathcal{P}$ stands for "batch arrivals with *precedence* constraints."

Definition 1: We say that the class of models $GI^{\mathcal{P}}/GI/N$ is potentially stable if there exists a queueing and service discipline that yields a stable queueing system (in the usual sense that the task queue length converges in distribution to a proper random variable).

Proposition 1: i) A necessary condition for the system $GI^{\mathcal{P}}/GI/N$ to be potentially stable is

$$\lambda E[C]E[B] \leq N$$

ii) A sufficient condition for the system $GI^{\mathcal{P}}/GI/N$ to be potentially stable is

$$\lambda E[C]E[B] < N.$$

Proof: i) The rate of arrival of tasks to the system shown in Fig. 4 is $\lambda E[C]$. Mean service time of each task is $E[B]$. If the system is stable, then the rate of task completion is $\lambda E[C]$. Applying Little's law to the N servers, we get mean number of busy servers = $\lambda E[C]E[B]$. Hence $N \geq \lambda E[C]E[B]$.

ii) We will show that there is a queueing and service discipline that renders the system stable under the given condition.

Given any precedence relation Γ, let $\hat{\Gamma}$ be a *linear* precedence relation (i.e., a chain or a complete order) such that $(i,j) \in \Gamma \Rightarrow (i,j) \in \hat{\Gamma}$. Given a job with precedence constraints Γ, if we process it according to $\hat{\Gamma}$ then none of the original constraints are violated and the task execution is sequential. Therefore we process jobs as follows: Replace the precedence graph Γ of a waiting job by a linear graph $\hat{\Gamma}$ as above. Whenever a complete job departs, start the first task of a waiting job on the free server. Continue processing this job on this server until all tasks are finished. It will be as if the job occupied the server for its total service time whose mean is $E[C]E[B]$. The system is now equivalent to a $GI/GI/N$ queue and hence the system is stable if $\lambda E[C]E[B] < N$ [33]. □

In the remainder of the paper we consider a special case of the class of models introduced in this section, namely, a fork-join queueing model with stochastic fork-join job structure.

III. Upper Bounds to the Mean Job Response Time: Single Fork-Join

Assume that the job arrival epochs form a Poisson process, and each job has only one fork-join. Thus, each job can be considered to be a batch of tasks. An arriving batch of tasks is split probabilistically into subbatches which are then assigned to the processing elements. Thus, the batch is *multinomially partitioned* over the N processors, each task going to the ith processor with probability p_i. Note that, in general the subbatches assigned to the queues are dependent, and some subbatches may be empty.

Thus, the job sojourn time is the maximum of the sojourn times of the nonempty subbatches, but these random variables are dependent. In this section we obtain upper bounds to the mean job sojourn time.

We use the following notation:

$B(t)$ - task service time distribution;
$\tilde{b}(s)$ - Laplace–Stieltjies Transform (LST) of the task service time distribution;
C_n - number of tasks in the nth arriving job;
$c(j) = P(C_n = j),\ n \geq 1,\ j \geq 0$;
$C_n^{(i)}$ - size of the nth arriving subbatch to processor i, $1 \leq i \leq N$;
$c^{(i)}(j) = P\{C_n^{(i)} = j\},\ 1 \leq i \leq N,\ j \geq 0,\ n \geq 1$;
$D_n^{(i)}$ - service time of the nth arriving subbatch to processor i, $1 \leq i \leq N$;
$\tilde{d}^{(i)}(s)$ - LST of the subbatch service time at the ith queue.

Note that, for analytical convenience we allow $C(0) = 0$, owing to Poisson job arrivals this does not loose any generality. Further, we assume that $B(0^+) = 0$; thus, a nonempty subbatch assigned to a queue has positive sojourn time with probability one.

We assume that the condition of stability for each queue is satisfied, i.e., for each queue, we have λ(mean subbatch service time) < 1 which gives the condition

$$\lambda\Big(\sum_{j=0}^{\infty} P\{C_n^{(i)} = j\}j\Big)(-\tilde{b}'(0)) < 1. \tag{1}$$

Let the stationary joint distribution of the virtual waiting times of the N queues be the same as that of the random vector $(W^{(1)}, \ldots, W^{(N)})$. $W^{(i)}$ will be the stationary distribution of the ith queue in isolation. This is simply an M/G/1 queue with batch arrivals, and the distribution of $W^{(i)}$ can be obtained as the waiting time distribution of an M/G/1 queue with an arrival rate of λ and whose service time LST is $\tilde{d}^{(i)}(s)$. Consider now a job arrival. Since job arrivals are Poisson, it follows from the PASTA (Poisson Arrivals See Time Averages) theorem [33] that the job sees the virtual waiting time vector $(W^{(1)}, W^{(2)}, \cdots, W^{(N)})$.

Thus, we have

$W^{(i)}$ - Stationary Waiting Time of the subbatch at the ith queue, $i = 1, 2, \cdots, N$;

Let $D^{(i)}$ denote a random variable with the same distribution as the common distribution of $D_n^{(i)}$, $n \geq 1$.

Hence, denoting the stationary sojourn time of a job by T, we have

$$T = \max_{1 \leq i \leq N} ((W^{(i)} + D^{(i)})I_{\{C^{(i)} > 0\}}) \tag{2}$$

where $I_{\{.\}}$ is the indicator function of the event $\{.\}$.

Recall that for analytical convenience, we allow empty batches to occur with positive probability. Thus, along with our assumption that task service times do not have an atom at 0, (2) implies that the atom of T at 0 corresponds to empty arriving batches. It follows that the mean sojourn time of the nonempty batches, i.e., the actual jobs, is $E[T]/P(C > 0)$. We shall seek bounds for $E[T]$ from which bounds for the mean job sojourn time will follow.

Owing to the fact that $D_n^{(i)} = 0$ iff $C_n^{(i)} = 0$, (2) can be written as

$$T = \max_{1 \leq i \leq N} (W^{(i)} + D^{(i)})I_{\{D^{(i)} > 0\}}. \tag{3}$$

Owing to independence between jobs, $(D^{(1)}, D^{(2)}, \cdots, D^{(N)})$ $\amalg$ $(W^{(1)}, W^{(2)}, \cdots, W^{(N)})$, where the symbol $\amalg$ stands for independence. We note here the fact that the $D^{(i)}$'s *are not independent* except for Poisson batch sizes.

A. A Simple Upper Bound

We use a lemma given by Aven [1] to bound the expression in (3). Let r be an integer such that $1 \leq r < \infty$, and X_i's are real valued random variables, then it has been proved [1] that

$$E[\max_{1 \leq i \leq N} X_i] \leq \{\sum_{i=1}^{N} E(|X_i|^r)\}^{\frac{1}{r}}. \tag{4}$$

Applying this to (3) for the job response time we get

$$\begin{aligned} E[T] &= E[\max_{1 \leq i \leq N}((W^{(i)} + D^{(i)})I_{\{D^{(i)} > 0\}})] \\ &\leq \{\sum_{i=1}^{N} E((W^{(i)} + D^{(i)})I_{\{D^{(i)} > 0\}})^r\}^{\frac{1}{r}}. \end{aligned}$$

We now attempt to simplify the expression for the upper bound above as follows: first consider the term $E(.)^r$, where, for ease of notation, we drop the superscript (i) from W and D. Then,

$$\begin{aligned} E[(W + D)^r] &= E[(W + D)^r I_{\{D=0\}}] \\ &\quad + E[(W + D)^r I_{\{D>0\}}] \\ &= E[W^r I_{\{D=0\}}] + E[(W + D)^r I_{\{D>0\}}] \\ &= E[W^r]P[D = 0] + E[(W + D)^r I_{\{D>0\}}] \end{aligned}$$

where the first term in last expression follows since W and D are independent random variables. We therefore have from above

$$E[(W + D)^r I_{\{D>0\}}] = E[(W + D)^r] - P(D = 0)E[W^r]. \tag{5}$$

Reintroducing the superscript (i), we have,

$$\begin{aligned} E[T] &\leq \{\sum_{i=1}^{N} E((W^{(i)} + D^{(i)})I_{\{D^{(i)} > 0\}})^r\}^{\frac{1}{r}} \\ &= \{\sum_{i=1}^{N} E[(W^{(i)} + D^{(i)})^r I_{\{D^{(i)} > 0\}}]\}^{\frac{1}{r}} \\ &= \{\sum_{i=1}^{N} (E[(W^{(i)} + D^{(i)})^r] \\ &\qquad - P(D^{(i)} = 0)E[(W^{(i)})^r])\}^{\frac{1}{r}}. \end{aligned}$$

Theorem 1: In the fork-join queue with Poisson job arrivals, i.i.d. task batches and i.i.d. task service times, an upper bound to the mean response time of the fork-join job is given by

$$\{\sum_{i=1}^{N}(E[(W^{(i)}+D^{(i)})^r]-P(D^{(i)}=0)E[(W^{(i)})^r])\}^{\frac{1}{r}} /P(C>0). \quad (6)$$

Proof: As above. □

Note that the result holds for any allocation policy for which the sequence $\{D_n^{(i)}, n \geq 1\}$ is i.i.d., for every $1 \leq i \leq N$.

In order to compute the upper bound, we require moments of the subbatch waiting time random variable W and the subbatch service time random variable D. Each of the N queues is an M/G/1 queue with batch arrivals at rate λ. The moments of the waiting time can be obtained from the Takács recurrence formula [13]. We have done the computations for multinomial partitioning of the task batches with $p_i = 1/N$.

The mean batch size ($E[C]$) is set equal to the number of processors, N. The mean task service time (μ^{-1}) is set equal to 1. When the task allocation probability p_i is equal for all the N queues, i.e., when $p_i = 1/N$, it follows that the service utilization factor at each queue (ρ) is equal to the job arrival rate (λ), since, $\rho = \lambda E[C]p_i/\mu$. We have studied Poisson and Geometric batch sizes. Exact value of the mean job sojourn time is obtained by simulation of the fork-join queue. Numerical results from these analysis will be presented later in this paper (see Figs. 5, 6, 7, and 8).

Owing to the obvious relation to the l_r norm, we call this the l_r upper bound. Note that for each case, r can be chosen to minimize the upper bound. Even with r chosen to optimize the bound, these upper bounds were found to be quite loose. Hence we search for better bounds.

B. Upper Bound Based on Associated Random Variables

We use the properties of *associated random variables* ([7], [4], [9]) to compute upper bounds to the mean job response time. The definition of associated random variables, and their useful properties, are given in Appendix A.

The statistics of the maximum of $\{T_1, \cdots, T_N\}$ are typically very difficult to compute in the presence of inter-variable *correlations*, as it is the case in many stochastic models of interest. However, when the RV's $\{T_1, \cdots, T_N\}$ are *associated*, Proposition A.2 (Appendix A) suggests a natural way of generating *computable bounds* on these statistics. This is the technique that has been adopted in the previous literature.

Thus, if $\{T_1, \cdots, T_N\}$ are associated, then

$$P[\max_{1\leq i\leq N} T_i > t] \leq 1 - \prod_{i=1}^{N} P(T_i \leq t). \quad (7)$$

It follows that if the random variables T_i, $i = 1, \cdots, N$ are identically distributed, then

$$E[\max_{1\leq i\leq N} T_i] \leq \int_0^\infty (1-(P(T_1 \leq t))^N)dt. \quad (8)$$

We now prove that *if the subbatches at the N queues are associated then the subbatch service times are also associated.* It will then follow that the *subbatch sojourn times* are *associated.* We will then be able to compute an *associativity upper bound* to the mean job response time.

As per the notation described before, we have the following:

$C^{(i)}$ - size of the subbatch at the ith processor, $i = 1, \cdots, N$

$D^{(i)}$ - service time of the subbatch at the ith processor, $i = 1, \cdots, N$

Let $Y_k^{(i)}$, $k \geq 1$ be i.i.d nonnegative random variables $\forall\ i, 1 \leq i \leq N$, *independent* of $\{C^{(1)},C^{(2)},\cdots,C^{(N)}\}$ and $\{Y_k^{(i_1)}, k \geq 1\}$ II $\{Y_k^{(i_2)}, k \geq 1\}$ $\forall\ i_1, i_2$. Let the distribution of $Y_k^{(i)}$ be $B(.)$, the task service time distribution.

Let

$$D^{(i)} = \sum_{k=1}^{C^{(i)}} Y_k^{(i)}, \quad 1 \leq i \leq N.$$

If $C^{(i)} = 0$ then the sum is taken to be zero.

Lemma 1: If the random variables $\{C^{(1)},C^{(2)},\cdots,C^{(N)}\}$ are *associated* then $\{D^{(1)},D^{(2)},\cdots,D^{(N)}\}$ are *associated.*

Proof: See Appendix B. □

Let $X^{(i)}, i = 1, 2, \cdots, N$ denote the subbatch sojourn time in the ith processor. Following our earlier notation for subbatch waiting time and subbatch service time, we have

Lemma 2: The random variables $X^{(i)}, i = 1, 2, \cdots, N$ given by $X^{(i)} = W^{(i)} + D^{(i)}$ are associated.

Proof: See Appendix B. □.

We have thus shown that if the subbatch sizes are associated then the subbatch sojourn times at the N queues are associated.

We now proceed to find situations in which subbatch sizes are associated.

First, we obtain a necessary condition. For simplicity, we consider the two processor model. Now, $C = C^{(1)} + C^{(2)}$, and the conditional joint distribution of $(C^{(1)}, C^{(2)})$ given C is Multinomial$(C, p, (1-p))$ where p is the probability of allocating a task to processor 1.

Clearly, if the random variables $C^{(1)}$ and $C^{(2)}$ are associated then [7]

$$Cov(C^{(1)}, C^{(2)}) \geq 0. \quad (9)$$

We shall use the notation $U \stackrel{\text{dist}}{=} V$ to denote that the random variables U and V have the same distribution.

Lemma 3: Let $C = C^{(1)}+C^{(2)}$, where $(C^{(1)}, C^{(2)})/C \stackrel{\text{dist}}{=} Multinomial(C, p, 1-p)$. Then, $Cov(C^{(1)}, C^{(2)}) \geq 0$ if and only if $Var(C) \geq E(C)$.

Proof: See Appendix B. □

It is easily verified that the condition $Var(C) \geq E(C)$ is satisfied if C is either Poisson, or Geometric or Negative Binomial. For Poisson, of course, $Var(C) = E(C)$.

It is clear that if the task batch size is Poisson distributed then the subbatches obtained by a multinomial partition are associated, by virtue of their being independent.

We now show that *multinomial partitions of a geometrically distributed random variable are associated.* It will then follow, from the arguments above, that the subbatch service times in the N queues are associated and finally the subbatch sojourn

times in the N queues are associated. Associativity upper bounds to the mean job response time then follow as discussed.

We begin by considering two processors. $C^{(1)}$ and $C^{(2)}$ are subbatch size random variables at the two queues respectively. C is the batch size random variable having a *geometric* distribution with parameter α. Hence,

$$C = C^{(1)} + C^{(2)}$$

$$C \stackrel{\text{dist}}{=} Geometric(\alpha)\ i.e.,$$

$$P(C = x) = \alpha^x(1-\alpha), x \geq 0,\ 0 \leq \alpha \leq 1$$

$$(C^{(1)}, C^{(2)})/C \stackrel{\text{dist}}{=} Multinomial(C, p, 1-p).$$

Thus, the *joint probability mass function* of $C^{(1)}$ and $C^{(2)}$ is given by

$$P(C^{(1)} = s, C^{(2)} = t) = \alpha^{s+t}(1-\alpha)\binom{s+t}{s}p^s(1-p)^t \stackrel{\text{def}}{=} f(s,t). \quad (10)$$

Proposition 2: $f(s,t)$ is totally positive of order 2.

Proof: See Appendix B. □

Corollary 1: The random variables $C^{(1)}$ and $C^{(2)}$ are associated.

Proof: This follows directly from Theorem 4.2 in [7] which states that if $C^{(1)}$ and $C^{(2)}$ are totally positive of order 2, then they are associated. □

We now extend the above results to our model with N processors, where a job is *multinomially partitioned* over the N processors, each task going to the ith processor with probability p_i.

As before, we have the random variables

C denoting batch size, and

$C^{(i)}$ denoting the subbatch at the ith procesor, $i = 1, \cdots, N$.

Using Corollary 4.15 in [7], the Corollary to our Proposition 2 is easily extended to N processors. This is done as follows. The batch size C has a geometric distribution with parameter α.

$$C \stackrel{\text{dist}}{=} Geometric(\alpha)$$

$$C = \sum_{i=1}^{N} C^{(i)}$$

$$(C^{(1)}, C^{(2)}, \cdots, C^{(N)})/C \stackrel{\text{dist}}{=} Multinomial(C, p_1, p_2, \cdots, p_N).$$

We denote the *joint probability mass function* of the $C^{(1)},\cdots,C^{(N)}$ by $f(x_1, \cdots, x_N)$. Then,

$$f(x_1, x_2, \cdots, x_N) = P(C^{(1)} = x_1, C^{(2)} = x_2, \cdots, C^{(N)} = x_N) = \alpha^{(x_1+x_2+\cdots+x_N)}(1-\alpha)\frac{(\sum_{i=1}^{N} x_i)!}{\prod_{i=1}^{N} x_i!}\prod_{i=1}^{N} p_i^{x_i}. \quad (11)$$

It can be shown quite easily that $f(x_1, \cdots, x_N)$ is totally positive of order 2 in each pair of arguments for fixed values of the remaining arguments. It follows from Corollary 4.15 in [7] that $\{C^{(1)},C^{(2)},\cdots,C^{(N)}\}$ are associated random variables.

We have thus proved

Theorem 2: The components of a Multinomial partition of a Geometric random variable are associated.

Computation of the upper bound is now merely a matter of computing the subbatch sojourn time distribution at each queue, and then using (8). Division by $P(C > 0)$ yields the desired upper bound. Laplace transforms, a numerical inversion technique, and numerical integration were used in carrying out the computations.

C. Comparison of the Two Upper Bounds with Simulation Results

We plot the mean job response time versus ρ (the server occupancy for each server) for Poisson and geometric batch size distributions. We consider two values for N, the number of processors, i.e., 4 and 7. The arrivals of jobs is Poisson and the task service times are assumed to be exponential with parameter 1. We have kept the mean job size equal to the number of processors (N) in the system. Figs. 5 and 6 show the mean job response time obtained through *simulation*, and the two upper bounds: the l_r *upper bound* and the just discussed *associativity upper bound* for Poisson batch size. Figs. 7 and 8 correspond to geometric batch size. In these figures only the *associativity* upper bound is shown. We see from these graphs that the l_r upper bounds are very loose as compared to the associativity upper bounds. Since the associativity upper bounds are quite close to the mean job response time obtained from the simulation study, they can be taken as a good approximation to the mean job response time.

The simulation programs have been written in SIMSCRIPT II.5. They were terminated after the variations in the measured mean response time were judged to be negligible.

Two further observations can be made from these numerical results. In making these, we recall that the variance to mean ratio for the geometric distribution is greater than 1, whereas for the Poisson distribution it is equal to 1. Thus, in this sense the geometric batch sizes display more variability than Poisson batch sizes. The results show that the mean job sojourn times with geometric batch sizes are larger than with Poisson batch sizes. Further, the associativity upper bound is looser for geometric batch sizes. This is because the positive correlation between subbatch sojourn times is accentuated by the positively correlated subbatches, thus making the independence approximation yield a looser bound.

IV. Lower Bounds to the Mean Job Response Time: Single Fork-Join

We continue with the same stochastic assumptions as in Section III.

A. Each Component Queue as a Lower Bound

From (3) and (5) it easily follows that

$$E[T] \geq \max_{1 \leq i \leq N}\{(1 - P(D^{(i)} = 0))E[W^{(i)}] + E[D^{(i)}]\} \quad (12)$$

where $E[W]$ can be found from standard analysis for the M/G/1 queue.

It is quite clear that this will yield a bound that is only a function of λ (or ρ) and not of N, the number of processors.

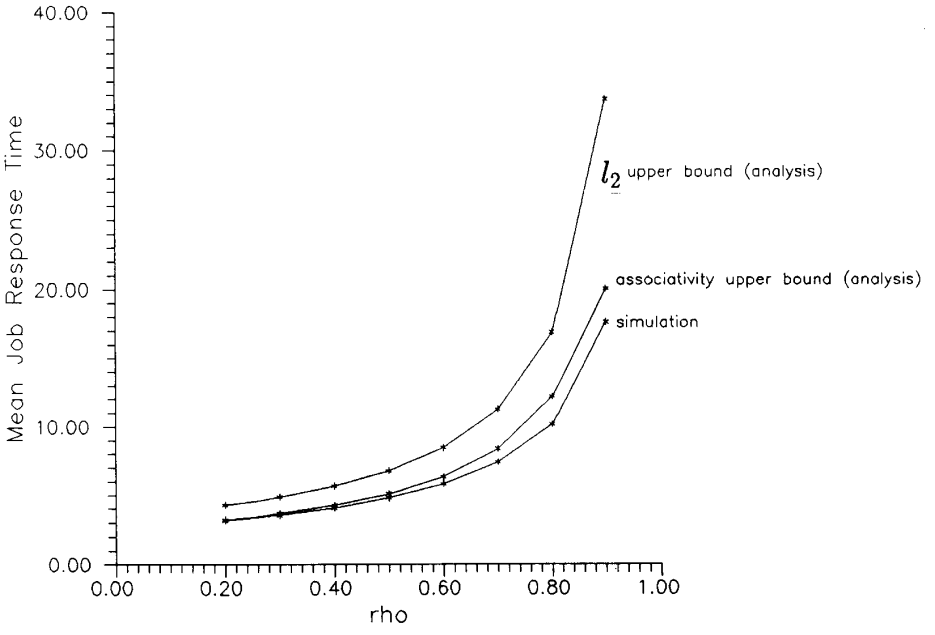


Fig. 5. Upper bounds on mean job response time. Number of processors = 4. Poisson batch size.

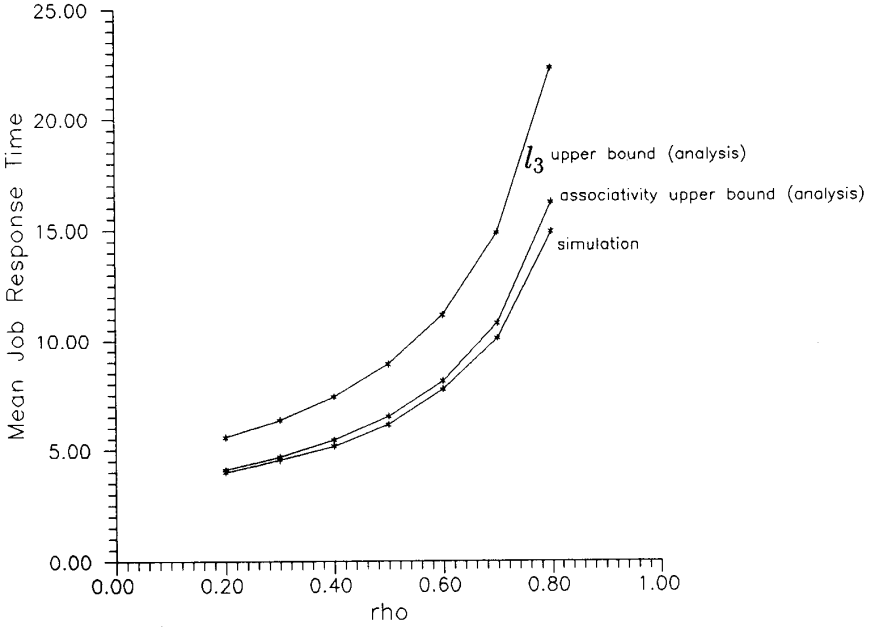


Fig. 6. Upper bounds on mean job respone time. Number of processors = 7. Poisson batch size.

Hence, this is not a useful lower bound. We therefore search for lower bounds which are a function of N.

We call this lower bound as *lower bound 1*.

B. Lower Bounds Neglecting Queueing Delays

From (3) we observe that, since $W^{(i)} \geq 0$,

$$T \geq \max_{1 \leq i \leq N} (D^{(i)} I_{\{D^{(i)} > 0\}}). \tag{13}$$

Now, define

$$\hat{T} = \max_{1 \leq i \leq N} (D^{(i)} I_{\{D^{(i)} > 0\}}). \tag{14}$$

Consider now a queueing system with a single queue that is served in a FCFS manner by the N servers. This system is shown in Fig. 9. Consider a single batch arriving to this system and finding the system empty. Let $\hat{T}'$ denote the sojourn time

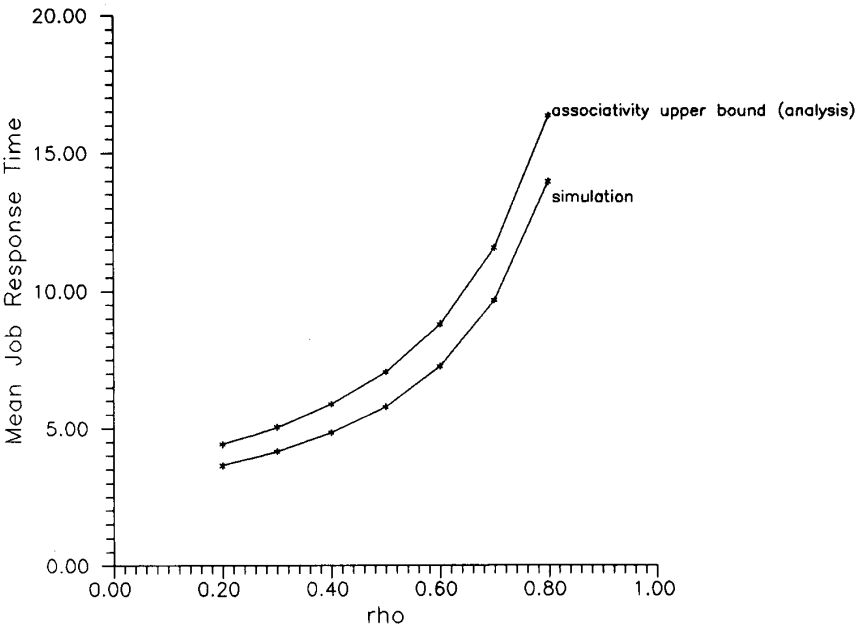


Fig. 7. Upper bounds on mean job response time. Number of processsors = 4. Geometric batch size.

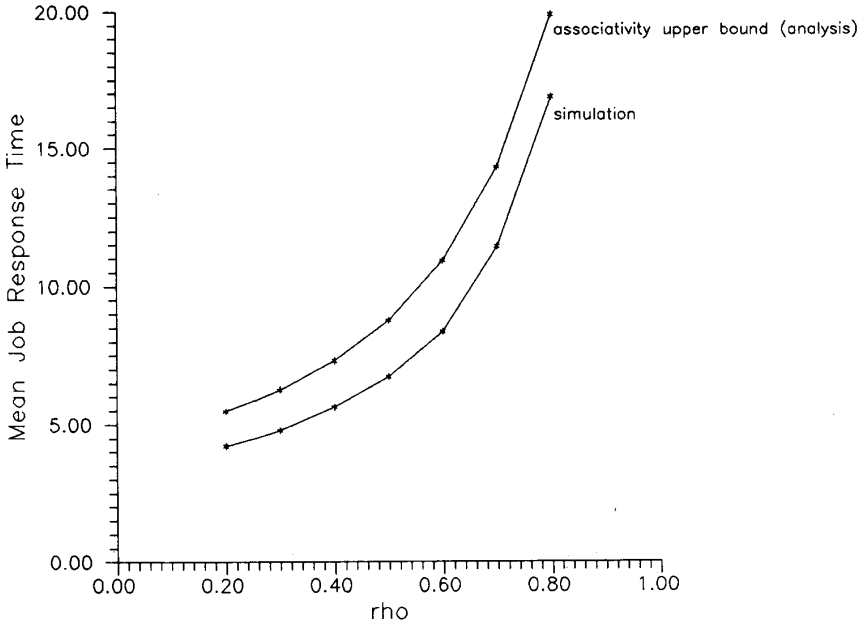


Fig. 8. Upper bounds on mean job response time. Number of processor = 7. Geometric batch size.

of this batch in the system. Denoting stochastic ordering by $\geq_{st}$, we have

Theorem 3:

$$\hat{T} \geq_{st} \hat{T}'$$

Proof: The result follows easily from the arguments in [34] and [27]. For consider a batch of size k entering either system (i.e., our original system and the system with a single queue). Denote by δ_i (resp., δ_i'), $1 \leq i \leq k$, the ith *ordered* departure epoch from the original system (resp., the single queue system). Clearly, for this batch of size k,

$$\hat{T} = \delta_k, \quad and$$
$$\hat{T}' = \delta_k'.$$

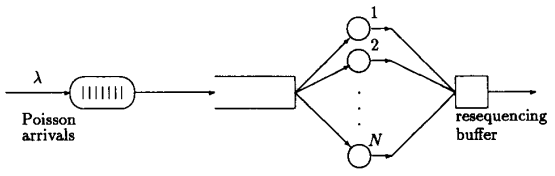


Fig. 9. The centralized system serving fork-join jobs.

But Wolff's argument [34] asserts that $\delta_k \geq_{\text{st}} \delta'_k$. It follows that (unconditioning on the batch size) $\hat{T} \geq_{\text{st}} \hat{T}'$. □

It follows that $E[T] \geq E[\hat{T}] \geq E[\hat{T}']$, hence $E[\hat{T}']$ is a lower bound to $E[T]$.

Notice that the above lower bound amounts to comparing the two queueing systems in the limit as $\rho \to 0$, and thus *ignores the queueing delays*. Unlike the bound in Section V-A, however, this bound does depend on the number of processors (N). We call this lower bound as *lower bound 2*.

For exponential task service time with mean $1/\mu$, it easily follows that

$$E[\hat{T}'] = \sum_{k=0}^{N} c(k)\frac{H_k}{\mu} + \sum_{k=N+1}^{\infty} c(k)\left(\frac{k-N}{N\mu} + \frac{H_N}{\mu}\right) \quad (15)$$

where

$$H_k = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{k}. \quad (16)$$

As before, we divide the lower bound $E[\hat{T}']$ by $(1-P(C=0))$ to get a lower bound on the actual delay of nonempty jobs.

The lower bound found above (*lower bound 2*) is a function of N only. We still seek a lower bound which varies with both ρ and N.

C. The Centralized System as a Lower Bound

In this section, we use the technique in [6] to show that if the task service time distribution is replaced by a *deterministic* distribution, having the same mean, then the resulting system is a lower bound in the sense of *convex increasing order*. Further, if in the latter system the N queues are replaced by a single queue serviced by the N servers, then we get a sample path wise lower bound that yields a lower bound to the original system in the sense of *convex increasing order*. The latter system is analyzable exactly.

In the first part, we prove that the original system with *deterministic* service time is a lower bound (in the sense of convex increasing ordering) on the same system with *exponential* service time. The proof is based on the properties of *convex increasing ordering* [33].

In the symbols we define below, subscript n refers to the nth arriving batch, $n \geq 0$. For the original system (with exponential task service times), as before, we recall the following definitions, where $1 \leq i \leq N$, and $n \geq 1$,

$D_n^{(i)}$ - service time of the subbatch at the ith queue;
$W_n^{(i)}$ - waiting time of the subbatch at the ith queue;
$T_n^{(i)}$ - sojourn time of the subbatch at the ith queue;
T_n - job sojourn time in the system;
b - mean task service time.

Let $\tilde{D}_n^{(i)}$, $\tilde{W}_n^{(i)}$, $\tilde{T}_n^{(i)}$, and $\tilde{T}_n$ be the corresponding quantities for the system with deterministic task service time b. The fixed service time is equal to the mean service time in our original system with exponential servers.

Finally, recall that,

τ_n - interarrival time between the nth and the $(n+1)$th job;
$C_n^{(i)}$ - subbatch size at the ith queue.

Lemma 4: For all $n \geq 0$, we have $\tilde{T}_n \leq_{\text{ci}} T_n$

Proof: See Appendix B for an outline of the proof. □

Corollary 2: If limiting distributions exist in either case, let T and $\tilde{T}$ denote the limiting random variables for the sojourn times in the two systems, then, $\tilde{T} \leq_{\text{ci}} T$.

Proof: Suppose, $\{\tilde{T}_n\}$ converges in distribution to $\tilde{T}$ and $\{T_n\}$ converges in distribution to T. If $E(\tilde{T})$ and $E(T)$ are finite, then, from Lemma 4 and from the property that $\leq_{\text{ci}}$ ordering is closed under convergence in distribution [33], we conclude that $\tilde{T} \leq_{\text{ci}} T$. □

Now, consider yet another system in which tasks with deterministic service times are all put into a single queue, that is served in a first-come-first-served fashion by the N servers. This represents a system with centralized queueing of tasks and distributed service [19]. Let $\tilde{T}'$ denote the stationary sojourn time in the centralized system. It is easily seen that (see [34], and [15]) the task sojourn times in the deterministic service time system with centralized queueing are smaller sample-path-wise than in the deterministic service time system with distributed queueing. These arguments and theorem immediately yield the following theorem

Theorem 4: $\tilde{T}' \leq_{\text{ci}} T$.

Thus, $E[\tilde{T}'] \leq E[T]$. We call the resulting lower bound *lower bound 3*.

The centralized system with deterministic service time is exactly analyzable for Geometric batch sizes (see Appendix C). Numerical results were obtained from this analysis. Exact analysis of the lower bound system for Poisson batches is difficult. For Poisson batches, however, the subbatches, allocated to queues are independent, and the lower bounds in [3] suffice.

D. Comparison of the Lower Bounds

We plot the three lower bounds to the mean response time of a fork-join job. As before we assume Poisson job arrivals and exponential task service times with a mean of 1. $E[T]$ and its bounds are plotted as a function of ρ, the occupancy of each server.

Fig. 10 corresponds to *Geometric batch size* and 4 processors ($N = 4$), whereas Fig. 11 corresponds to *Geometric batch size* with 7 processors ($N = 7$).

We find that for this set of parameters values, lower bound 1 performs the best. As expected, lower bound 2, which is not a function of load, performs the worst. Note, however, that as N increases, if we let the mean task batch size be equal to N, lower bound 3 will increase and eventually exceed lower bound 1. In fact, if the task batch size is equal to N, then for geometric batch size, lower bound 3 at $\rho = 0$ has a limit of $b/(1-e^{-1})$ as $N \to \infty$, whereas lower bound 1 at $\rho = 0$ is equal to b for all N. Note also that at $\rho = 0$ lower bound

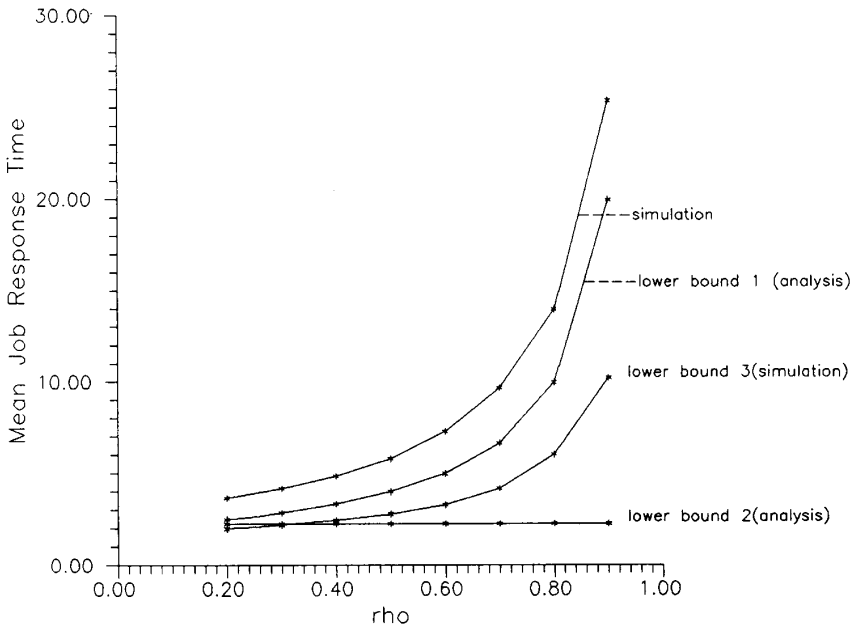


Fig. 10. Lower bounds to the mean job response time. Number of processors = 4. Geometric batch size.

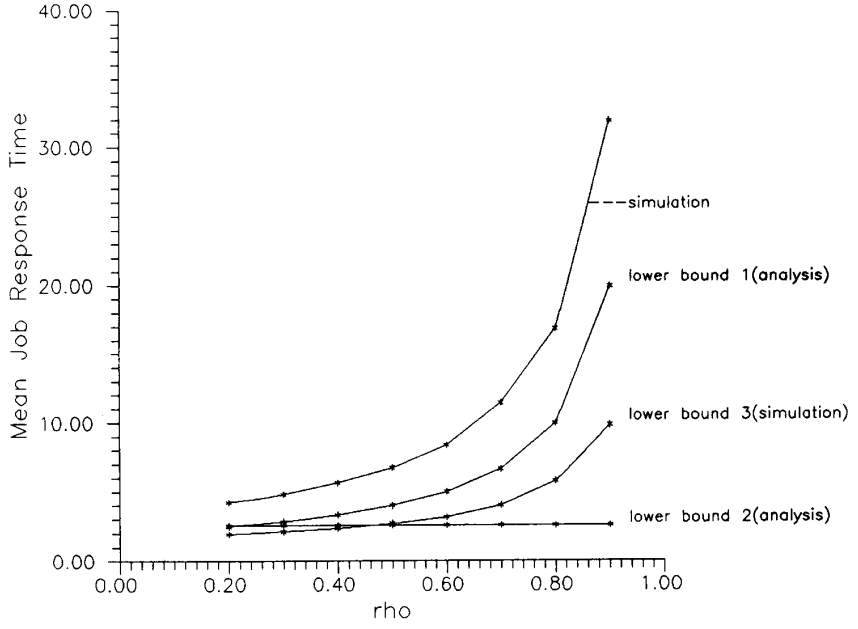


Fig. 11. Lower bounds to the mean job response time. Number of processors = 7. Geometric batch size.

2 will always exceed lower bound 3, since lower bound 3 corresponds to deterministic service times.

V. Jobs With Multiple Fork-Joins

In this section, we study a simple approximation for the mean sojourn time of jobs with multiple fork-joins. The number of fork-joins in a job is a random variable. A job, thus, has a random number of fork-joins, each fork-join with a random number of tasks (see Fig. 1).

A. Model Description

We model the situation with jobs that have multiple fork-joins as a fork-join N processor system with *feedback* (see Fig. 12). After the completion of a join, with probability p, the job

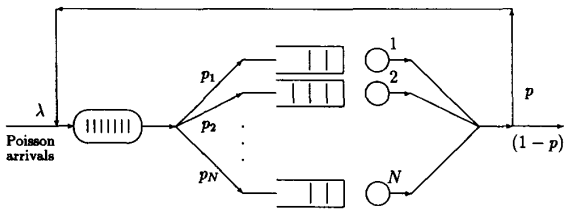


Fig. 12. A multiple fork-join queueing system.

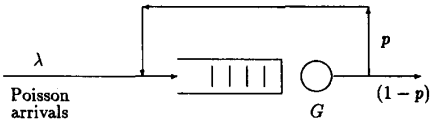


Fig. 13. M/G/1 queue with feedback.

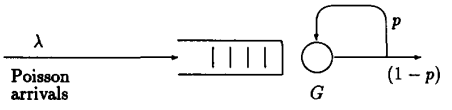


Fig. 14. M/G/1 queue with feedback at the server.

is fedback to the main arrival stream, and with probability $(1-p)$ the job leaves the system. Thus, the number of fork-joins in a job is *geometrically distributed* with parameter p, i.e.,

$$P(K=k) = p^{k-1}(1-p), \quad k \geq 1, \quad 0 \leq p \leq 1 \tag{17}$$

where K is the random variable corresponding to the number of fork-joins in the job.

Job arrivals are *Poisson* with rate λ. The number of tasks in a single fork-join has a *Poisson* distribution. Further, we assume that the task service times are *exponentially* distributed with parameter μ.

As compared to the simple fork-join system, the analysis of the fork-join queueing system with feedback is considerably harder. Note that although the external arrival process of jobs is Poisson, owing to feedback, the aggregate arrival process of fork-joins to the queue is not Poisson.

B. An Approximation to the Mean Job Response Time

To motivate the approximate analysis of a fork-join queueing system with feedback, we first recall the analysis of an M/G/1 queue with feedback. This is shown in Fig. 13, where G denotes the service time random variable.

Single task jobs arrive at the M/G/1 queue with rate λ. With probability p, a customer is fed back to the queue and with probability $(1-p)$ the customer departs from the queue. This queue is equivalent to an M/G/1 queue with feedback at the server itself in the sense that the *queue length distribution* is the same. An M/G/1 queue with feedback at the server is shown in Fig. 14 with feedback probability p. Fig. 15 shows the same queue with the effective service time random variable denoted by B. We note that B is just a geometric sum of G's.

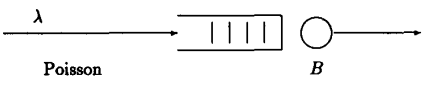


Fig. 15. An equivalent M/G/1 queue.

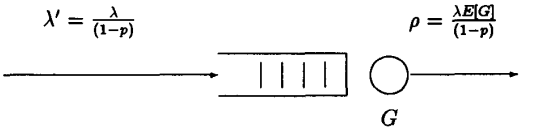


Fig. 16. An approximation to M/G/1 queue with feedback.

The two queueing systems shown in Fig. 13 and Fig. 15 are equivalent in terms of the mean delay per job.

We cannot extend the above idea of feedback at the server to the fork-join queueing system with feedback. This is because of the join synchronization delays experienced by a subbatch that finishes earlier than the other subbatches in a batch. These subbatches must "join" with their other "siblings" before being fed back, thus rendering instantaneous feedback an inappropriate model.

We reconsider, therefore, the M/G/1 queue with feedback. Let us denote the net arrival rate of jobs at the queue in Fig. 13 by λ'. It is clear that $\lambda' = \lambda/(1-p)$, provided $\lambda/(1-p) < (E[G])^{-1}$ (i.e., the rate of arrivals of fork-joins in the original model is λ').

A simple approximation for the M/G/1 queue with feedback would be to consider an M/G/1 queue with arrival rate λ' and service time distribution G (see Fig. 16), and then approximate the job sojourn time in the feedback model with the convolution of a geometric number of sojourn times in the model of Fig. 16.

Letting W' denote the approximation to the mean waiting time obtained this way, we will get

$$E[W'] = \frac{\rho}{1-\rho}\left(\frac{c_G^2+1}{2}\right)\frac{E[G]}{(1-p)} \tag{18}$$

where ρ is the service utilization factor and c_G^2 is the coefficient of variation of the service time G, given by,

$$c_G^2 = \frac{Var(G)}{(E[G])^2}. \tag{19}$$

Let W be the waiting time random variable in the M/G/1 queue with feedback at the server (i.e., the *exact* waiting time), as shown in Fig. 13.

Proposition 3: $E[W'] \geq E[W]$ if $c_G^2 \geq 1$.

Proof: In Fig. 15, B is the service time RV which is equal to a *geometric* sum of G's. From (17) the LST of B can then be written as

$$\begin{aligned}\tilde{b}(s) &= \sum_{k=1}^{\infty} p^{k-1}(1-p)[\tilde{g}(s)]^k \\ &= \frac{(1-p)\tilde{g}(s)}{1-p\tilde{g}(s)}.\end{aligned} \tag{20}$$

The second moment of the service time random variable B can be found from the expression (20) as follows:

$$E[B^2] = \tilde{b}''(s)\Big|_{s=0} = \frac{E[G^2](1-p) + 2p(E[G])^2}{(1-p)^2}. \quad (21)$$

For $E[W']$ to be an *upper bound* to $E[W]$, the following inequality must be satisfied:

$$\begin{aligned} &\frac{E[G^2]}{(1-p)^2} \geq E[B^2] \\ \Longrightarrow\ &\frac{E[G^2]}{(1-p)^2} \geq \frac{E[G^2](1-p) + 2p(E[G])^2}{(1-p)^2} \\ \Longrightarrow\ &\frac{Var(G)}{(E[G])^2} \geq 1 \\ \Longrightarrow\ &c_G^2 \geq 1. \end{aligned} \quad (22)$$

Therefore, the inequality (22) implies that the mean waiting time in the M/G/1 queue in Fig. 16 is an *upper bound* to the mean waiting time of the M/G/1 queue shown in Fig. 15. □

We use this simple analysis to motivate the following approximation.

The aggregate fork-join arrival rate is $\lambda/(1-p)$. The approximation takes this process of arrivals of fork-joins to be Poisson.

Let $\hat{T}$ denote the response time of a fork-join in this system.

We have already computed upper bounds to $E[T]$ in Section IV. These were based on the concept of *associated random variables*.

Let $\hat{S}_K$ denote the approximate response time of a job with K fork-joins. Then,

$$E[\hat{S}_K] = E(\sum_{i=1}^{K} \hat{T}_i) \quad (23)$$

where $\hat{T}_i$ is the approximate response time of the ith fork-join within a job. In steady state, the $\hat{T}_i$'s are identically distributed. Thus, $E[\hat{T}_i] = E[\hat{T}]\ \forall\ i$.

Since K is independent of $\hat{T}_i$'s, it follows from *Wald's lemma* [33] that

$$E(\sum_{i=1}^{K} \hat{T}_i) = E(K)E(\hat{T}). \quad (24)$$

It follows from (17) that $E[K] = 1/(1-p)$.

Finally, since we consider only nonempty jobs, i.e., those in which there is at least one nonempty fork-join, we have

$$E[\hat{S}_K] = \frac{E(K)E(\hat{T})}{P(nonempty\ job)} = \frac{E(\hat{T})}{(1-p)P(nonempty\ job)}. \quad (25)$$

We have assumed the distribution of the number of tasks in a fork-join to be Poisson with parameter x. We now find the probability of a nonempty job.

$$\begin{aligned} P(nonempty\ job) &= 1 - P(all\ subjobs\ are\ empty) \\ &= 1 - \sum_{n=1}^{\infty} P(K=n)[P(empty\ job)]^n \\ &= 1 - \sum_{n=1}^{\infty} p^{n-1}(1-p)(e^{-x})^n \\ &= \frac{1-e^{-x}}{1-pe^{-x}}. \end{aligned} \quad (26)$$

From (26) and (25), we get the expression for $E[\hat{S}_K]$ given as

$$E[\hat{S}_K] = \frac{E[\hat{T}](1-pe^{-x})}{(1-p)(1-e^{-x})}. \quad (27)$$

We now consider a single queue in the fork-join queueing system with feedback. This is an M/G/1 queue with *bulk arrivals* and *feedback*, with a synchronization delay after the service of a subbatch. Note that for **Poisson** batch size, the size of a subbatch at each component queue is also Poisson distributed. It is easily shown that the coefficient of variation of the service time random variable, G, given by c_G^2, satisfies the inequality $c_G^2 \geq 1$. This condition along with Proposition 3 suggests that the approximation to the response time of a job ($E[\hat{S}_K]$) may be an *upper bound* to $E[S_K]$ (the actual response time of a job).

We use the associativity upper bound for $E[\hat{T}]$, and in each case use the corresponding value of $E[\hat{S}_K]$ as an approximation to the mean sojourn time of a job with K fork-joins.

C. Numerical Results

Denote by S_K the actual response time random variable of a job with multiple fork-joins. In Table I, we show the approximation to the mean response time of the job obtained using (27). The table shows approximation and simulation results for various values of the number of processors (N), the arrival rate of jobs (λ), and the parameter of the geometric distribution of the number of fork-joins in a job (p).

The distribution of the fork-join batch size is Poisson with parameter (mean) x. The task service time is an exponential random variable with parameter 1. Finally, ρ is the occupancy of each processor.

We see that for $\rho \leq 0.6$ the approximation to the mean response time of the job is good. The value of $E[S_K]$ is obtained from *simulation*. Notice that, as suggested by the analysis earlier, the approximation *over estimates* $E[S_K]$. Recall that there are actually two approximations being made here: the Poisson arrival of the successive fork-joins of a job, and the approximation to the mean delay of each fork-join. It has been mentioned before that due to *feedback*, the Poisson arrival nature of the fork-joins is destroyed. This is because, an arrival at the queueing system triggers another arrival, thereby the *interarrival times* become dependent. However, in the case of a multiple fork-join queue, there is a synchronization delay incurred at the join. This added delay makes the arrivals to the system less bursty and the Poisson assumption may actually be a good approximation. Thus, most of the error in the approximation may be due to the approximation for $E[T]$.

TABLE I
APPROXIMATION TO $E[S_K]$ IN MULTIPLE FORK-JOIN QUEUE

N	p	λ	ρ	$E[S_K]$	*Approximation*
4	0.3	0.20	0.286	4.881	5.126
		0.50	0.714	10.811	12.528
		0.60	0.857	19.648	22.802
	0.5	0.20	0.400	8.249	8.469
	0.7	0.23	0.766	29.818	35.105
	0.9	0.05	0.500	36.484	50.228
5	0.3	0.20	0.286	5.464	5.690
		0.50	0.714	12.242	13.920
		0.60	0.857	23.559	25.107
7	0.3	0.20	0.286	6.292	6.599
		0.50	0.714	14.613	16.169
		0.60	0.857	27.376	28.658

VI. COMPARISON OF TWO JSQ ALLOCATION POLICIES

In this section we study two versions of the Join-the-Shortest-Queue (JSQ) policy for single fork-join jobs: *JSQ by batch* and *JSQ by task*. In JSQ by batch, an arriving batch (with a random number of tasks) is assigned to the shortest of the N queues in the multicomputer system. We therefore refer to this as task allocation with *no splitting*. In JSQ by task, the tasks of an arriving batch are ordered in an arbitrary way and are then successively assigned to the shortest queue as if they were a stream of task arrivals. This queueing system is thus *JSQ with splitting*.

Through simulation and analysis of these two JSQ allocation policies, we see that for low and moderate utilizations of the queues, JSQ by task yields a strictly lower value of mean job response time as compared to JSQ by batch. However, at high utilizations, the two policies approach each other in mean delay performance. Thus, if communication delays were included in the model, JSQ by batch would yield *lower mean response time* than JSQ by task for high loads. This suggests an adaptive allocation policy: do JSQ by task at low and moderate loads and do JSQ by batch at high loads.

A. Simulation Results

For the simulation study, we keep the batch size distribution and the number of processors fixed and plot the mean job response time ($E[T]$) versus ρ (the occupancy of each server) for the two systems: JSQ with splitting and JSQ with no splitting. In the simulation study, job arrivals are *Poisson*. Job size distribution is *Poisson* with mean batch size being equal to the number of processors. Task service times are *exponentially* distributed with parameter set to 1.

Figs. 17 and 18 show the comparison of the two systems, with $N = 4$ and 7, respectively. From the figures, one sees that JSQ with splitting yields a lower expected response time for low to medium utilizations but as we go to high utilizations, JSQ by task (splitting) and JSQ by batch (nonsplitting) yield almost the same response time.

Further, it is seen from the curves that as we go from $N = 4$ to $N = 7$, the point at which the two curves meet, shifts toward higher ρ values.

Nelson [18] makes a similar observation, but the two queueing systems are slightly different from ours. They compare two different multiprocessor architectures, one in which fork-join jobs are executed concurrently by all the processors and the other where these jobs are executed sequentially and shortest queue routing is used. The job size is deterministic and is equal to the number of procesors. In the splitting model, upon a job arrival, the N tasks are scheduled so that one goes to each of the N processors. The nonsplitting model is the same as ours, except that, in [18], the number of tasks in a job is equal to N. The comparison of the two systems in [18] reveals a tradeoff between them. The splitting model has a lower expected response time for low to medium utilizations but for high utilizations the nonsplitting model is better. The crossover occurs because JSQ is not used when the job is split over several processors.

Thus, our simulation results and those in [18] suggest the following: for low uilizations, executing the tasks of a job concurrently is beneficial, whereas for large utilizations, nonsplitting of jobs yields a lower response time.

B. Analytic Explanations of the Results

We now give analytical explanations for our simulation results. For low utilizations, it can be proved that JSQ by task yields a strictly lower value of mean job response time than that in JSQ by batch.

Consider the case when $\rho \to 0$. For such a system, there is no queueing and we can consider a single batch arriving at an empty queueing system. Now, in *JSQ by batch*, the entire batch is allocated to one queue whereas in *JSQ by task*, tasks within a batch are allocated successively into the shortest queue as if they were separate arrivals. In [32], it is proved that of all the allocation strategies, the strategy that minimizes the mean customer waiting time is the one which assigns each arrival to the *shortest queue*, provided each customer's service time is a random variable with a nondecreasing hazard rate [7].

Since, in our model, task service time is exponential, which corresponds to a constant hazard rate (hence, nondecreasing), it follows from the proof in [32] that JSQ by task yields lower mean job response time as compared to JSQ by batch in the limit when $\rho \to 0$, i.e., at low load values.

Theorem 5: In the limit when $\rho \to 0$, JSQ by task yields a strictly lower value of mean job response time as compared to JSQ by batch.

Proof: As discussed above. □

At high utilizations, we see that the two policies approach each other in mean delay performance. This can be explained as follows. In the JSQ by task policy, the queue lengths at the N queues of the system are stochastically equal under heavy loads. This in turn means that under heavy loads, the synchronization delays in this system are negligible. In JSQ by batch, we have seen that there are no synchronization delays, as the entire batch is assigned to the shortest queue. Further, heavy traffic analysis shows that the task queue lengths in the two systems become stochastically the same under heavy loads. Thus, the subbatch delays in JSQ by task policy, and the batch delay in the JSQ by batch policy tend to become equal under heavy loads.

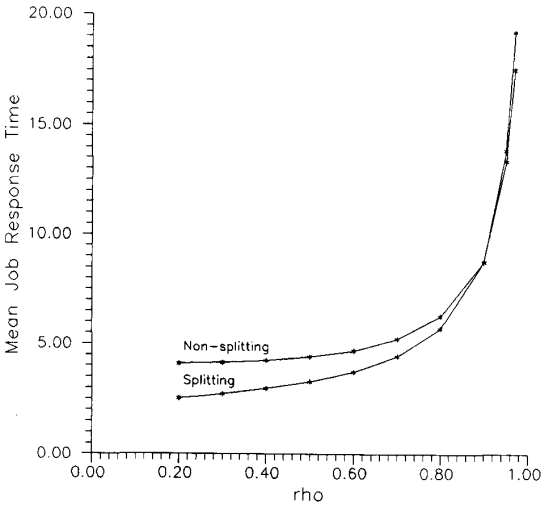


Fig. 17. JSQ allocation (Poisson batch size). Comparison of JSQ by task (splitting) versus JSQ by batch (nonsplitting). $N = 4$.

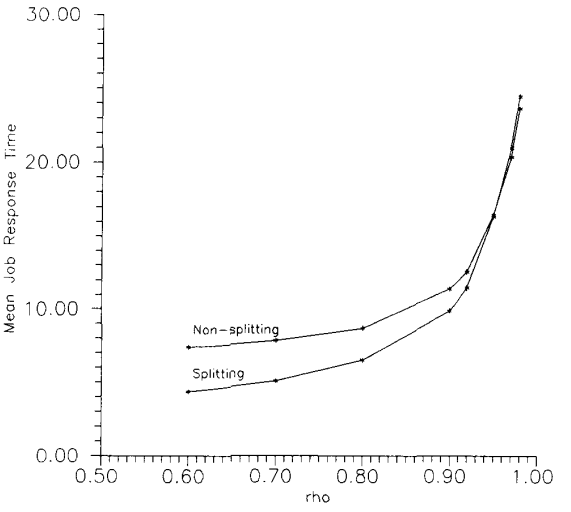


Fig. 18. JSQ allocation (Poisson batch size). Comparison of JSQ by task (splitting) versus JSQ batch (nonsplitting) $N = 7$.

In fact using some diffusion approximation results [22], the following explicit results can be developed. Consider Geometric batch sizes, with $C(k) = x^{k-1}(1-x)$, $k \geq 1$, $0 < x < 1$, and a Poisson job arrival process. It follows that the task arrival process is also renewal. Assume exponential task service times. The foregoing implies that the batch service times are also i.i.d. exponential. It can now be shown, using the results in [22], that in the heavy traffic limit

1) the normalized job sojourn time in JSQ by batch, and the normalized job sojourn time in JSQ by task, converge weakly to the same process, and
2) the normalized task sojourn time and batch sojourn time in JSQ by task converge to the same heavy traffic limit. This last result is obtained via a simple modification of the proof of Theorem 3 in Section 5.2 of [22].

VII. Discussion

We have studied the performance of a distributed computing system where each processor has its own queue of tasks, and incoming jobs to the system consist of batches of tasks with precedence constraints among them defined by a linear sequence of forks and joins. The main points of deviation from other works are: a) using a distributed system of processors each with its own queue as opposed to a central queue b) using a random number of tasks generated by a fork and (c) using a random number of forks and joins in a given job. As in most other works in the literature, communication and coordination overheads have been neglected.

To study the above class of systems, we have extended the scope of fork-join queueing models by allowing a random number of tasks in each fork, different task allocation policies, and multiple forks in a job. Most of the earlier work in fork-join queueing models assumed a deterministic structure of fork-join jobs. Although, Nelson *et al.* [19], did consider fork-join jobs with a random number of tasks, however, they have analyzed a centralized parallel processing system, where the N identical processors serve a single queue.

What makes our model more complex to analyze than those studied previously is the fact that the job structure is stochastic and, in general, the task subbatches allocated to the N queues are dependent. This, in turn, causes the task subbatch service times to be dependent. This additional dependency between the N queues does not permit the analysis in the literature to be directly applicable. Moreover, in our model, since the number of tasks in a job is possibly different from the number of processors, a task allocation policy needs to be specified.

In Section II, we considered a general model of a parallel processing system and derived the stability conditions. The main result in Section II is a Proposition where we proved a necessary and sufficient condition for the queueing system to be *potentially* stable.

For the case of jobs with a single fork-join and, probabilistic allocation of tasks of the job to the N processors, we obtained upper and lower bounds to the mean response time of the fork-join job. We developed the upper bounds in Section III and the lower bounds in Section IV.

In Section III we proved (Lemma 1) that if the random variables corresponding to the task subbatch sizes at the N queues are associated, then the subbatch service times in the N queues are associated. It follows from this result that the subbatch sojourn times at the N queues are associated (Lemma 2). This allows us to find an upper bound to the mean response time of a fork-join job. We then went on to prove that if the task batch size distribution is geometric, the probabilistic partition of the batch yields associated subbatches (Theorem 2). This, combined with the previous result, yields an upper bound to the mean job sojourn time. By comparison with simulation, it was found that this upper bound serves as a reasonably good approximation. Simulation studies have shown that for the same mean batch size, geometric batches yield larger mean delays than Poisson batches (see [25]).

The main results in Section IV are Lemma 4 and Theorem 4. In Lemma 4, we showed that if the task service time distribution is replaced with a deterministic distribution, with the same mean, then the mean job response time in the resulting system is a lower bound in the sense of convex increasing order. In Theorem 4, we further prove that if in the latter system the N queues are replaced by a single queue serviced by the N servers, then we get a further lower bound in the sense of convex increasing order. This system is analyzable exactly for geometric batch sizes (Appendix C).

In Section V, for the case of multiple fork-join jobs we studied an approximation motivated by an approximation for the M/G/1 queue with feedback.

Finally, we discussed an interesting simulation result in Section VI, where we compared two JSQ (Join the Shortest Queue) allocation policies, namely, JSQ by task and JSQ by batch. We were able to prove analytically using earlier known results that in the limit when $\rho \to 0$, JSQ by task has smaller delay than JSQ by batch (Theorem 5). However, at high loads, the two policies begin to yield the same mean delay. This suggested that if the communication delays were to be incorporated, then JSQ by task would become worse at high loads, thus motivating an adaptive allocation policy: do JSQ by task at light loads and JSQ by batch at high loads.

There are several extensions of this work that are of interest. Extending the basic model to the case where the servers are of different speeds would model a system consisting of nonhomogeneous processors. Furthermore, one could relax the exponential assumptions on the arrival process.

We have not modeled communication delays in the fork-join queueing system. In practical systems such communication delays exist because of either interprocess communication or moving the tasks from one processor to another processor. Thus, incorporation of communication delays in the analysis of a fork-join queueing system will be a useful extension. Some related work may be found in [17] and [16].

The approximation for the multiple fork-join queueing system is rather ad hoc, and more work is needed to develop provable bounds, and good approximations.

Appendix A
Associated Random Variables

Definition: The $\Re$-valued random variables $\{T_1, T_2, \cdots, T_n\}$ are said to be *associated* if the inequality

$$E[f(T)g(T)] \geq E[f(T)]E[g(T)] \tag{28}$$

holds for all monotone nondecreasing mappings $f, g : \Re^n \to \Re$ for which the expectations $E[f(T)], E[g(T)]$ and $E[f(T)g(T)]$ exist.

Association of random variables satisfies the following desirable multivariate properties [7]:

- (P_1) Any subset of associated random variables is associated.
- (P_2) The set consisting of a single random variable is associated.
- (P_3) Increasing functions of associated random variables are associated.
- (P_4) If two sets of associated random variables are independent of one another, then their union is a set of associated random variables.
- (P_5) Independent random variables are associated.

The key result which we use in the computation of the upper bound to the mean job response time is given as a proposition below: [29]

Proposition A.1: If the random variables $\{T_1, T_2, \cdots, T_K\}$ are associated, then the inequalities

$$P[T_i \leq a_i, 1 \leq i \leq K] \geq \prod_{i=1}^{K} P[T_i \leq a_i] \tag{29}$$

hold true for all $a = (a_1, \cdots, a_K)$ in $\Re^K$.

An alternate and useful way of expressing (29) is obtained as follows: The $\Re$-valued random variables $\{\bar{T}_1, \cdots, \bar{T}_K\}$ are said to form an *independent version* of the random variables $\{T_1, \cdots, T_K\}$ if

1) The random variables $\{\bar{T}_1, \cdots, \bar{T}_K\}$ are *mutually independent*, and
2) For every $1 \leq i \leq K$, the random variables T_i and $\bar{T}_i$ have the *same* probability distribution.

With this notion, Proposition A.1 takes the following form. [29]

Proposition A.2: If the random variables $\{T_1, \cdots, T_K\}$ are associated, then the inequalities

$$P[T_i \leq a_i, 1 \leq i \leq K] \geq P[\bar{T}_i \leq a_i, 1 \leq i \leq K] \tag{30}$$

hold true for all a_i in $\Re^K$, whence

$$\max_{1 \leq i \leq K} T_i \leq_{\text{st}} \max_{1 \leq i \leq K} \bar{T}_i. \tag{31}$$

Definition: [7] Let S and T be two *discrete* random variables. The *joint frequency function* of S and T is denoted by $f(s,t)$ and is defined as

$$f(s,t) \stackrel{\text{def}}{=} P(S = s, T = t).$$

We then say that $f(s,t)$ is *totally positive of order* 2 if

$$\begin{vmatrix} f(s_1,t_1) & f(s_1,t_2) \\ f(s_2,t_1) & f(s_2,t_2) \end{vmatrix} \geq 0$$

for all $s_1 < s_2$, $t_1 < t_2$ in the domain of S and T.

Appendix B
Proofs

Lemma 1: If the random variables $\{C^{(1)}, C^{(2)}, \cdots, C^{(N)}\}$ are *associated* then $\{D^{(1)}, D^{(2)}, \cdots, D^{(N)}\}$ are *associated.*

Proof: Since $Y_k^{(i)}$, $1 \leq i \leq N$, $k \geq 1$ are mutually independent random variables, therefore they are associated. Further, since $\{C^{(1)}, C^{(2)}, \cdots, C^{(N)}\}$ are associated and $\{C^{(1)}, C^{(2)}, \cdots, C^{(N)}\}$ II $\{Y_k^{(i)}, 1 \leq i \leq N, k \geq 1\}$, therefore it follows from property P_4 of associated random variables that $\{C^{(1)}, C^{(2)}, \cdots, C^{(N)}, Y_k^{(i)}, 1 \leq i \leq N, k \geq 1\}$ are associated.

Now $D^{(i)} = \sum_{k=1}^{C^{(i)}} Y_k^{(i)}$, $1 \leq i \leq N$. It follows that $D^{(i)}$ can be written as a function f of $C^{(i)}, Y_k^{(i)}$, $k \geq 1$ or $D^{(i)} = f(C^{(i)}, Y_1^{(i)}, Y_2^{(i)}, \cdots)$ where $Y_k^{(i)} \geq 0$, $k \geq 1$. We

observe that f is nondecreasing in each argument. Hence by property P_3, the RV's $\{D^{(1)},D^{(2)},\ldots,D^{(N)}\}$ are associated since each is an increasing function of associated RV's. □

Lemma 2: The random variables $X^{(i)}, i = 1,2,\cdots,N$ given by $X^{(i)} = W^{(i)} + D^{(i)}$, are associated.

Proof: In the previous lemma, we have shown that the $D^{(i)}$, $i = 1,2,\cdots,N$ are associated.

Denote

$W_n^{(i)}$ - waiting time of the nth subbatch to processor $i, 1 \leq i \leq N$;

$D_n^{(i)}$ - service time of the nth subbatch to processor $i, 1 \leq i \leq N$;

$\tau_{n+1}^{(i)}$ - Interarrival time between the nth and the $(n+1)$th subbatch to queue i.

We have

$$X_n^{(i)} = W_n^{(i)} + D_n^{(i)}. \tag{32}$$

We first establish that $W_n^{(i)}, i = 1,2,\cdots,N$ are *associated.* We prove this by *induction* following an argument essentially the same as in [20].

Basis Step: The random variables $W_1^{(i)}, i = 1,2,\cdots,N$ are associated since, assuming that the system is initially empty, the equations $W_1^{(i)} = 0, i = 1,2,\cdots,N$, hold *with probability 1.*

Inductive Step: Assume that $W_m^{(i)}, i = 1,2,\cdots,N$, are associated for $m = 1,2,\cdots,n$. We will show that this implies that $W_{n+1}^{(i)}, i = 1,2,\cdots,N$ are associated. We have the following equality [6]

$$W_{n+1}^{(i)} = (W_n^{(i)} + D_n^{(i)} - \tau_{n+1}^{(i)})_+, \qquad i = 1,2,\cdots,N \tag{33}$$

where $(x)_+ = \max(0,x)$.

We note that the max function is an increasing function. By the *inductive hypothesis,* $W_n^{(i)}, i = 1,2,\cdots,N$ are associated. The $D_n^{(i)}$'s, $i = 1,\cdots,N$ are associated, by Lemma 1. We observe that

$$\tau_{n+1}^{(i)} = \tau_{n+1} \tag{34}$$

is one random variable, *independent* of all others. Hence, the random variable $-\tau_{n+1}$ is associated.

Therefore, $W_{n+1}^{(i)}, i = 1,2,\cdots,N$ are associated, since (33) shows that they are increasing functions of associated random variables.

The union of sets $W_n^{(i)}, i = 1,\cdots,N$ and $D_n^{(i)}, i = 1,\cdots,N$ are associated. It follows that since $X_n^{(i)}, i = 1,\cdots,N$, is an increasing function of associated RV's, it is associated.

Finally, assuming that each of the N queues is *stable* and *ergodic,* the stationary limits $X^{(i)}, i = 1,\cdots,N$ are also associated. □

Lemma 3: Let $C = C^{(1)} + C^{(2)}$, where $(C^{(1)},C^{(2)})/C \overset{\text{dist}}{=} Multinomial(C,p,1-p)$. Then $Cov(C^{(1)},C^{(2)}) \geq 0$ if and only if $var(C) \geq E(C)$.

Proof: Clearly $C^{(1)}/C \overset{\text{dist}}{=} Binomial(C,p)$. Denoting by $\tilde{c}(.)$ and $\tilde{c}^{(i)}(.)$ the generating functions of C and $C^{(i)}$, it easily follows that $\tilde{c}^{(1)}(z) = \tilde{c}(pz + (1-p))$. Hence

$$E[C^{(1)}] = pE[C]$$
$$E[C^{(1)}]^2 = p^2E[C^2] + p(1-p)E[C].$$

Further,

$$E[C^{(1)}C^{(2)}] = E[C^{(1)}(C - C^{(1)})] = E[CC^{(1)}] - E[C^{(1)}]^2. \tag{35}$$

Now,

$$E[CC^{(1)}] = E\{E[CC^{(1)}/C]\} = E\{CE[C^{(1)}/C]\} = E(pC^2) = pE[C^2].$$

Hence,

$$E[C^{(1)}C^{(2)}] = pE[C] - pE[C] - p(1-p)E[C], \tag{36}$$

and

$$E[C^{(1)}]E[C^{(2)}] = E[C^{(1)}]E[C - C^{(1)}] = p(1-p)(E[C])^2 \tag{37}$$

Thus

$$E[C^{(1)}C^{(2)}] - E[C^{(1)}]E[C^{(2)}] = \{p(1-p)E[C^2] - p(1-p)E[C]\} - p(1-p)(E[C])^2 \tag{38}$$

Hence $E[C^{(1)}C^{(2)}] \geq E[C^{(1)}]E[C^{(2)}]$ iff

$$p(1-p)E[C^2] - p(1-p)E[C] \geq p(1-p)(E[C])^2. \tag{39}$$

This is always true if $p = 1$ or $p = 0$, whereas, if $0 < p < 1$, it is true iff the following holds:

$$E[C^2] - E[C] \geq (E[C])^2 \Leftrightarrow \frac{Var(C)}{E[C]} \geq 1. \tag{40}$$

Therefore, we have for $0 < p < 1$,

$$Cov(C^{(1)},C^{(2)}) \geq 0 \Longleftrightarrow Var(C) \geq E[C] \tag{41}$$

which proves the lemma. □

Proposition 2: $f(s,t)$ is totally positive of order 2.

Proof: With $s_1 < s_2$ and $t_1 < t_2$ consider the inequality

$$\begin{vmatrix} f(s_1,t_1) & f(s_1,t_2) \\ f(s_2,t_1) & f(s_2,t_2) \end{vmatrix} \geq 0. \tag{42}$$

This is true iff

$$\alpha^{s_1+t_1}(1-\alpha)\binom{s_1+t_1}{s_1}p^{s_1}(1-p)^{t_1}\alpha^{s_2+t_2}(1-\alpha)\binom{s_2+t_2}{s_2} \cdot p^{s_2}(1-p)^{t_2} - \alpha^{s_1+t_2}(1-\alpha)\binom{s_1+t_2}{s_1}p^{s_1}(1-p)^{t_2}\alpha^{s_2+t_1} \cdot (1-\alpha)\binom{s_2+t_1}{s_2}p^{s_2}(1-p)^{t_1} \geq 0$$

$$\iff \binom{s_1+t_1}{s_1}\binom{s_2+t_2}{s_2} - \binom{s_1+t_2}{s_1}\binom{s_2+t_1}{s_2} \geq 0$$
$$\iff \frac{(s_1+t_1)!}{s_1!t_1!}\frac{(s_2+t_2)!}{s_2!t_2!} - \frac{(s_1+t_2)!}{s_1!t_2!}\frac{(s_2+t_1)!}{s_2!t_1!} \geq 0$$
$$\iff (s_1+t_1)!(s_2+t_2)! - (s_1+t_2)!(s_2+t_1)! \geq 0.$$

Since $s_1 < s_2$ and $t_1 < t_2$, the above expression can be written as

$$(s_1+t_1)!\{(s_2+t_2)(s_2+t_2-1)\dots(s_2+t_1+1)\}(s_2+t_1)! - \{(s_1+t_2)(s_1+t_2-1)\dots(s_1+t_1+1)\}(s_1+t_1)!(s_2+t_1)! \geq 0.$$

$$\iff (s_2+t_2)(s_2+t_2-1)\cdots(s_2+t_1+1) - (s_1+t_2)(s_1+t_2-1)\cdots(s_1+t_1+1) \geq 0. \quad (43)$$

The last inequality holds since $s_2 > s_1$. Hence $f(s,t)$ is totally positive of order 2, which proves the proposition. □

Lemma 4: For all $n \geq 0$, we have $\tilde{T}_n \leq_{\text{ci}} T_n$.

Proof: We have that

$$T_n = \max_{1\leq i\leq N}\{W_n^{(i)} + D_n^{(i)}\}I_{\{C_n^{(i)}>0\}}$$

and that

$$\tilde{T}_n = \max_{1\leq i\leq N}\{\tilde{W}_n^{(i)} + \tilde{D}_n^{(i)}\}I_{\{C_n^{(i)}>0\}}.$$

Denote by τ_n, $n \geq 0$, the inter arrival time between the nth and the $(n+1)$th batch. Let $\mathcal{G}$ be the sigma field defined by

$$\mathcal{G} = \sigma\{(C_1^{(1)}, \cdots, C_1^{(N)}, \tau_1), (C_2^{(1)}, \cdots, C_2^{(N)}, \tau_2), \cdots\}.$$

Where $\sigma\{\cdot\}$ as usual denotes the σ field generator by the random variables in $\{\cdot\}$.

Now clearly, for $1 \leq i \leq N$, $n \geq 0$,

$$E(D_n^{(i)}/\mathcal{G}) = b\, C_n^{(i)} = \tilde{D}_n^{(i)}. \quad (44)$$

It now easily follows from an argument identical to that in [6], that for $1 \leq i \leq N$, and $n \geq 0$,

$$E(W_n^{(i)}/\mathcal{G}) \geq \tilde{W}_n^{(i)}.$$

Now

$$T_n = \max_{1\leq i\leq N}(W_n^{(i)} + D_n^{(i)})I_{\{C_n^{(i)}>0\}}.$$

Hence, by the convexity of $\max\{\}$

$$E(T_n/\mathcal{G}) \geq \max_{1\leq i\leq N} E(((W_n^{(i)} + D_N^{(i)})I_{\{C_n^{(i)}>0\}})/\mathcal{G}).$$

But $I_{\{C_n^{(i)}>0\}} \in \mathcal{G}$.

Hence

$$E(T_n/\mathcal{G}) \geq \max_{1\leq i\leq N} E(((W_n^{(i)} + D_n^{(i)})/\mathcal{G})\, I_{\{C_n^{(i)}>0\}})$$
$$\geq \max_{1\leq i\leq N}(\tilde{W}_n^{(i)} + \tilde{D}_n^{(i)})\, I_{\{C_n^{(i)}>0\}}$$
$$= \tilde{T}_n.$$

It now follows that, for $n \geq 0$,

$$T_n \geq_{ci} \tilde{T}_n$$

□

Appendix C
Analysis of $M^X/D/N$ Queue With Batch Synchronization

We observe that $\tilde{T}'$ is the batch sojourn time in an $M^X/D/N$ queue in which a batch departs the system only when all its constituent tasks have completed service. We begin by noting that with deterministic service times, and FCFS service, the batch sojourn time in such a system is the same as the sojourn time of the last task in a batch.

We shall consider the case in which the batch size has a geometric distribution, i.e.,

$$c(k) = P(C = k) = x^{k-1}(1-x),\ k \geq 1.$$

Let the limiting random variable of the task sojourn time be denoted by $\tilde{S}'$. It then follows from a result [11] that $\tilde{S}'$ and $\tilde{T}'$ have the same distribution for geometrically distributed batch sizes. This is easily seen as follows. Order the tasks in the successive batches in some arbitrary way and let $\tilde{S}_i$, $i \geq 1$ denote the sojourn time of the ith task.

Then

$$E\tilde{S}' = \lim_{n\to\infty}\frac{1}{n}\sum_{i=1}^{n}\tilde{S}_i' \quad a.s.$$

Let

$$\ell_i = \begin{cases} 1 & \text{if the } i\text{th task is the last task in a batch} \\ 0 & \text{otherwise} \end{cases}.$$

Then

$$E\tilde{T}' = \lim_{n\to\infty}\frac{1}{\sum_{i=1}^{n}\ell_i}\sum_{i=1}^{n}\ell_i\tilde{S}_i' \quad a.s.$$

Note that for geometric batch sizes, $\{\ell_i\}$ is a Bernoulli sequence, hence by "Bernoulli Arrivals See Time Averages" [11], it follows that

$$E\tilde{S}' = E\tilde{T}'.$$

Observe that this result utilizes crucially the fact that service times are deterministic.

It now remains to determine the distribution of $\tilde{T}'$. The $M^X/D/N$ queue can be exactly analyzed using the well known Crommelin argument and matrix analytic methods [21]. The transition matrix of the embedded Markov chain, in the Crommelin approach has exactly the same structure as that in the Bailey Bulk Server Queue model.

Letting mean task service time $b = 1$, the vector $(a_0, a_1, \cdots, a_k, \cdots)$ is given by

$$a_0 = e^{-\lambda}$$

and for $k \geq 1$

$$a_k = P\{k \ task \ arrivals \ in \ unit \ time\}$$
$$= \sum_{i=1}^{k} \frac{\lambda^i e^{-\lambda}}{i!} P\{k \ task \ arrivals / \ i \ batch \ arrivals\}$$
$$= \sum_{i=1}^{k} \frac{\lambda^i e^{-\lambda}}{i!} \left\{ \binom{k-1}{i-1} (1-x)^i \, x^{k-i} \right\}.$$

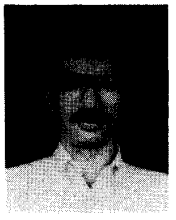
Observe that the term in $\{.\}$ in the last expression is the Negative binomial distribution.

ACKNOWLEDGMENT

We are grateful to Prof. L. M. Patnaik, Department of Computer Science and Automation, Indian Institute of Science, for useful discussions and for providing the computing facilities in his laboratory. Useful comments from anonymous referees have greatly improved the presentation of the paper.

REFERENCES

[1] T. Aven, "Upper(lower) bounds on the mean of the maximum(minimum) of a number of random variables," *J. Appl. Prob.*, vol. 22, pp. 723–728, 1985.
[2] F. Baccelli, "Two parallel queues created by arrivals with two demands: The M/G/2 symmetrical case," Report INRIA, no. 426, July 1985.
[3] F. Baccelli and A. Makowski, "Simple computable bounds for the fork-join queue," in *Proc. Conf. Inform. Sci. Syst.*, John Hopkins Univ., Baltimore, MD, Mar. 1985, pp. 436–441.
[4] ——, "Multidimentional stochastic ordering and associated random variables," *Oper. Res.*, vol. 37, no. 3, May-June, 1989, pp. 478–487.
[5] F. Baccelli, A. M. Makowski, and A. Shwartz, "Simple computable bounds and approximations for the fork-join queue," in *Proc. Int. Workshop Comput. Performance Evaluation*, Tokyo, Japan, Sept. 1985, pp. 437–450.
[6] F. Baccelli, W. A. Massey, and D. Towsley, "Acyclic fork-join queueing networks," *J. ACM*, vol. 36, no. 3, pp. 615–642, July 1989.
[7] R. E. Barlow and F. Proschan, *Statistical Theory Of Reliability and Life Testing: Probability Models.* New York: Holt, Rinehart and Winston, 1975.
[8] D. Bertsekas and R. Gallager, *Data Networks.* Englewood Cliffs, NJ: Prentice-Hall, 1987.
[9] J. D. Esary, F. Proschan, and D. W. Walkup, "Association of random variables, with applications," *Ann. Math. Statist.*, 38, pp. 1466–1474.
[10] L. Flatto and S. Hahn, "Two parallel queues created by arrivals with two demands," *SIAM. J. Appl. Math.*, vol. 44, pp. 1041–1053, 1984.
[11] S. Halfin, "Batch delays versus customer delays," Bell Labs. Tech. Memo., to be published.
[12] P. B. Hansen,"The programming language Concurrent Pascal," *IEEE Trans. Software Eng.*, vol. SE-1, pp. 199–207, June 1975.
[13] L. Kleinrock, *Queueing Systems, Vol. 1: Theory.* New York: Wiley, 1975.
[14] C. P. Kruskal and A. Weiss, "Allocating independent subtasks on parallel processors," *IEEE Trans. Software Eng.*, vol. SE-11, pp. 1001–1016, Oct. 1985,.
[15] J. Kuri and A. Kumar, "On the optimal allocation of customers that must depart in sequence," *Operations Res. Lett.*, to be published.
[16] ——, "Optimal control of arrivals to queues with delayed queue length information," in *Proc. 31st IEEE Conf. Decision and Control*, Tucson, AZ, Dec. 1992.
[17] R. Mirchandaney, D. Towsley, and J. A. Stankovic, "Analysis of the effects of delays on load sharing," *IEEE Trans. Comput.*, vol. 38, no. 11, pp. 1513–1525, Nov. 1989.
[18] R. D. Nelson and T. K. Philips, "An approximation for the mean response time for shortest queue routing with general interarrival and service times," Research Report, IBM, RC 15429(# 68659).
[19] R. Nelson, D. Towsley, and A. N. Tantawi,"Performance analysis of parallel processing systems," *IEEE Trans. Software Eng.*, vol. 14, no. 4, pp. 532–539, Apr. 1988.
[20] R. Nelson and A. N. Tantawi, "Approximate analysis of fork/join synchronization in parallel queues," *IEEE Trans. Comput.*, vol. 37, no. 6, pp. 739–743, June 1988.
[21] M. F. Neuts, "Queues solvable without Rouche's theorem," *Oper. Res.*, vol. 27, no. 4, July-Aug. 1979.
[22] M. Reiman, "Some diffusion approximations with state space collapse," in *Lecture Notes on Control and Information Science.*
[23] S. M. Ross, *Stochastic Processes.* New York: Wiley, 1983.
[24] E. C. Russell, Building Simulation Models with SIMSCRIPT II.5, CACI, 1983.
[25] R. Shorey, "Performance analysis and scheduling of stochastic fork-join jobs in a multicomputer system," M.Sc(Engineering) thesis, Dep. Elec. Commu. Eng., Indian Institute of Science, Bangalore, Oct., 1990.
[26] K. Sigman, "Queues as Harris Recurrent Markov Chains," *Queueing Syst.*, vol. 3, pp. 179–198, 1988.
[27] D. R. Smith and W. Whitt, "Resource sharing for efficiency in traffic systems," *Bell Syst. Tech. J.*, vol. 60, no. 1, Jan. 1981.
[28] D. Stoyan, *Comparison Methods for Queues and other Stochastic Models (English Translation),* D. J. Daley, Ed. New York: Wiley.
[29] Takagi, Ed., *Stochastic Analysis of Computer and Communication Systems.* Amsterdam: North-Holland, 1990, pp. 57–129.
[30] D. Towsley, J. A. Rommel, and J. A. Stankovic, "The performance of processor sharing scheduling fork-join in multiprocessors," in *High-Performance Computer Systems,* E. Gelenbe, Ed. Amsterdam: North-Holland, 1988, pp. 146–156.
[31] J. P. Tremblay and R. Manohar, *Discrete Mathematical Structures with Applications to Computer Science,* McGraw-Hill Computer Science Series, 1987.
[32] R. R. Weber, "On the optimal assignment of customers to parallel servers," *J. Appl. Prob.*, vol. 15, pp. 406–413, 1978.
[33] R. W. Wolff, *Stochastic Modeling and the Theory of Queues.* Englewood Cliffs, NJ: Prentice-Hall, 1989.
[34] ——, "An upper bound for multi-channel queues," *J. Appl. Prob.*, vol. 14, pp. 884–888, 1977.



Anurag Kumar (S'77–M'81–SM'92) was born on July 13, 1955 in Meerut, India. He received the B.Tech. degree in electrical engineering from Indian Institute of Technology, Kanpur, in 1977, and was awarded the President's Gold Medal. He then received the Ph.D. degree (major: electrical engineering, minor: OR/math) from Cornell University, Ithaca, NY, in January 1982.

From December 1981 to May 1988, he was with the Performance Analysis Department of AT&T Bell Laboratories, Holmdel, NJ, where he worked on the modeling and analysis of switching systems, survivable networks, data communication networks, inventory systems, and load controls of distributed systems. In June 1988 he joined the Department of Electrical Communication Engineering, Indian Institute of Science (IISc), Bangalore, India, where he is an Associate Professor, and is also a Coordinator of the Education & Research Network Project (ERNET). Currently, his general area of interest is communication networks; in particular his research interest is in stochastic modeling, analysis, optimization and control problems arising in communication networks.

Dr. Kumar is an Associate Member of Sigma Xi.

Rajeev Shorey was born in New Delhi, India, on January 28, 1963. He received the B.Sc. degree from St. Stephen's College, Delhi University, in 1984, and the B.E. degree in computer science and engineering from Indian Institute of Science, Bangalore, in 1987.

After working for one year in HCL Ltd. as a software engineer in the R & D computer division of the firm, he joined Indian Institute of Science, Bangalore, as a research scholar in 1988. In 1991, he received the M.Sc.(Engineering) degree from the Department of Electrical Communication Engineering, Indian Institute of Science. Currently, he is working towards the Ph.D. degree in the same department. His research interests include parallel and distributed computing, queueing theory, distributed discrete event simulation, performance analysis of computer systems and computer communication networks.

Mr. Shorey is a student member of the IEEE Computer Society.