Computing Performance Bounds for Fork-Join Queueing Models

John C.S. Lui[1]
Computer Science Department
The Chinese University of Hong Kong

Richard R. Muntz[2]
UCLA
Computer Science Department

Don Towsley[3]
University of Massachusetts
Computer Science Department

[1]The work of this author was supported in part by the Croucher Foundation and the Direct Research Grant from CUHK.
[2]The work of this author was supported in part by the National Science Foundation under grant CCR-9215064.
[3]The work of this author was supported in part by the National Science Foundation under grant NCR-9116183.

Abstract

We study a computer system which accepts parallel programs which can be modeled using the fork-join computational paradigm. The system under study has K homogeneous servers, each having an infinite capacity queue. Jobs arrive to the system according to a general interarrival process with mean arrival rate λ. Upon arrival, the job is split into K independent tasks $t_i, 1 \leq i \leq K$ and task t_i is assigned to the i^{th} server. Each task requires a mean service time of $1/\mu$. Each server uses the First-Come-First-Serve (FCFS) scheduling discipline to service its tasks. A job is complete upon the completion of its last task. This kind of queueing model has no known closed form solution in the general ($K \geq 2$) case. Rather than complete modification of the arrival and service distributions to obtain bounds on job response time on a fork-join queueing model as reported in previous literature, we show that by modifying the arrival and service distributions at some imbedded points in time, we can obtain better performance bounds. We also provide an efficient algorithm that can compute upper and lower bounds on the expected response time of jobs in a fork-join queueing model. The methodology presented allows one to tradeoff the tightness of the bounds and computational cost. Examples are presented which show the excellent relative accuracy achievable with modest computational cost.

1 Introduction

With the advent of multiprocessing technology, parallel programming languages [14, 26] and parallel programming environments [5, 11], there is an increasing interest in understanding and modeling the performance of parallel programs. In this paper, we propose a methodology to evaluate the performance of parallel programs which have a *fork-join* control structure on a parallel machine.

We model this kind of parallel program as follows. Assume the computing system has K homogeneous servers each with an infinite capacity queue. A parallel program (or job) with general interarrival distribution arrives to this parallel computing system with mean arrival rate λ. Upon arrival, the job splits into K independent tasks $t_i, 1 \leq i \leq K$ and task t_i is assigned to the i^{th} server which requires a mean service time of $1/\mu$. Each server uses the First-Come-First-Serve (FCFS) scheduling discipline to service its tasks. When a task is finished and if there are any tasks of the same job still in service, the finished task will wait in the synchronization area. The job is considered complete (and it departs from the system) when all its tasks have completed their service. This kind of parallel program paradigm arises in many application areas. For example, in a parallel database machine with a shared-nothing architecture [29], a complex query is partitioned so that there is a local query on each processing nodes. Another example is a computer vision system where an image is fed to a multiprocessor system, each processor does ray-tracing calculations for a portion of the rays. The final frame is ready for display when all processors have finished their ray-tracing operations.

Performance evaluation of this type of fork-join parallel program is difficult because:

- the arrival processes to the servers are correlated so the model cannot be decomposed.
- since each server has infinite queueing capacity, the state space of the system is multi-dimensional and infinite in each dimension.
- there is no known closed-form solution.
- it is impossible to use numerical methods directly since the state space of the problem is infinite.

We start with a brief review of the published literature on the analysis of fork-join queueing models. First, exact analysis is possible when the system is significantly simplified, for example, if we assume the job arrival process is Poisson with tasks having exponential service time distribution and the number of servers is equal to two. The exact analysis for this system can be found in [9, 10, 24]. Nelson and Tantawi [24] also proposed a scaling approximation technique for $K \geq 2$ homogeneous exponential servers. Recently, an extension to this approximation approach was developed by Makowski and Verma [22]. A more general model is considered in [1, 2, 3] where arrival and service processes are general. An upper bound is obtained by assuming K mutually independent $GI/G/1$ parallel queuing systems while the lower bound assumes $D/G/1$ parallel queueing systems. The tightness of these bounds is not investigated. Heidelberger and Trivedi [12] developed an approximate solution with bounds for a queueing network with jobs having computation graphs with a fork phase only (i.e., no synchronization of the completion of the tasks). In [13], the model is extended to contain a fork and join node and two approximation techniques are presented: one is based on a decomposition approach and other is based on an iterative solution method. Stability conditions for fork-join queueing networks were investigated by Konstantopoulos and Walrand [17] while the mean job response time was studied in [4] but the tightness of the bounds was not investigated. In [6], bounds on job response time were derived based on two-server models. Lastly, models have been investigated for programs exhibiting parallel fork-join structures that are executed on multiple servers sharing a single queue [18, 25, 30].

Rather than complete modification of the arrival and service distributions to obtain bounds on the mean job response time as reported in previous literature, we show that by modifying the arrival and service distributions at some *imbedded points* in time, we can obtain better performance bounds. We also provide an efficient algorithm that can compute upper and lower bounds on the expected response time of jobs. This methodology allows one to tradeoff the tightness of the bounds and computational cost. Examples are presented which show the excellent relative accuracy achievable with modest computational cost. Therefore, by providing much tighter as well as computable bounds, our results are distinguished from previous work on this problem.

The organization of the paper is as follows. In Section 2, we formally define the parallel program with fork-join paradigm that we are analyzing. In Sections 3 and 4, we present modified models; one gives an upper bound for mean job response time and the other provides a lower bound on job response time. We will illustrate how these modified models have a structure which yields an efficient numerical solution. Also, we will prove that these models do provide bounds and we will also present the algorithmic approach to computing these bounds. In Section 5, we present an application with some numerical examples. These examples illustrate how do we apply the model to compute the expected job response time for a parallel workstation cluster computing environment. Conclusions are given in Section 6.

2 Model Description

We consider a queueing system which consists of K identical servers each having an infinite capacity queue. The parallel program is modeled as an external job with interarrival distribution being a series-parallel phase type distribution where each phase is exponentially distributed [16]. The mean arrival rate of external jobs is denoted by λ. Upon arrival, a job forks into K tasks, namely $t_i, 1 \leq i \leq K$. Task t_i, which is assumed to have a k-stage Erlang distribution with mean service time requirement of $1/\mu$, is assigned to the i^{th} processor. Each processor is modeled as a single server with infinite queueing capacity and FCFS scheduling discipline. The system model is depicted in Figure 1. The stability condition for this kind of queueing system is [22]:

$$\lambda \;<\; \mu \tag{1}$$

which is the local stability condition for each server in the system. A job leaves the system as soon as all its tasks are completed. For a partially completed job, the completed tasks are forced to wait in the synchronization area. A job completes and departs when its last component task has been completed. We are interested in the expected job response time which we denote by T.

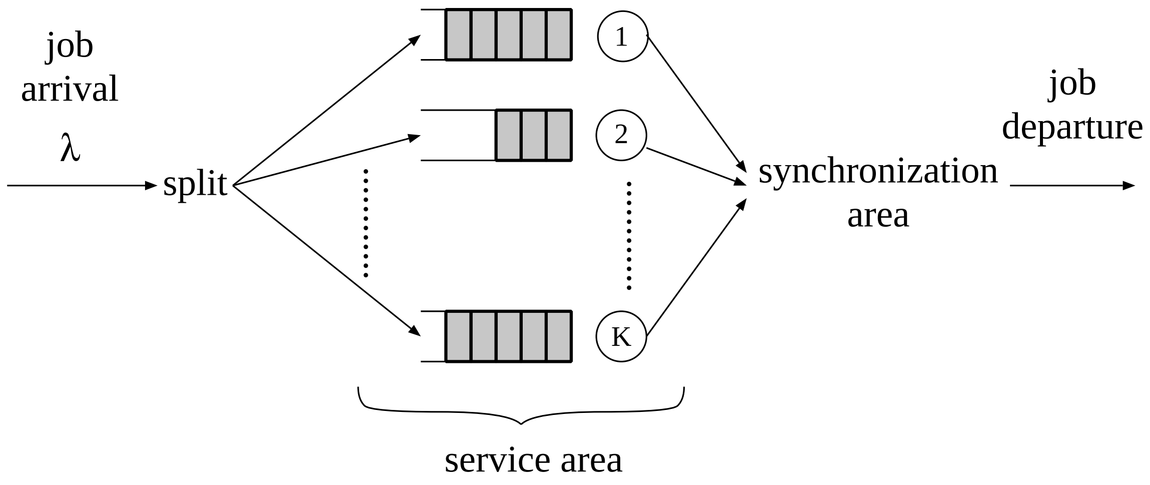

Figure 1: Parallel Program with Fork-Join Model.

Since the interarrival arrival process is a series-parallel phase type distribution, let there be $R \geq 1$ parallel phases and for phase i, it has $r_i \geq 1$ series stages. we can construct a Markov model, M, for this queueing system with state space description as:

$$\boldsymbol{N} \;=\; ([a,b],[N_1,p_1],\ldots,[N_K,p_K]) \qquad \text{for} \quad b \leq R,\, a \leq r_b,\, N_i \geq 0,\, 0 \leq p_i \leq k$$

where $[a, b]$ indicates the state of the series-parallel phase type distribution of the interarrival process[1], N_i is the number of tasks in the waiting queue of the i^{th} server, p_i indicates the servicing stage of the task in the i-th server. By convention, $p_i = 0$ when there is no task being serviced at server i. Given a state s, we can find the number of tasks waiting in the synchronization area as follows. Let $\hat{N}_i = N_i + 1\{p_i > 0\}$. $\hat{N}_i$ is the number of tasks in the i-th server since N_i is the number of waiting tasks at the i^{th} server and $1\{p_i > 0\}$ indicates whether there is any task in service at the i^{th} server. Define $N^* = \max(\hat{N}_1, \ldots, \hat{N}_K)$. Then the number of tasks that are waiting in the synchronization area is:

$$\textstyle\sum_{i=1}^{K}(N^* - \hat{N}_i) \tag{2}$$

The unique steady state probability vector for this continuous-time discrete state (CTDS) Markov model M satisfies the following system of linear equations:

$$\vec{\pi}G = \vec{0} \qquad and \qquad \vec{\pi}\underline{e} = 1 \tag{3}$$

where $\vec{\pi}$ is the K-dimensional steady state probability vector, $\underline{e}$ denotes an appropriately dimensioned column vector of $1's$ and G is the transition rate matrix. having the following transition structure.

If the event is a transition in the phase of the interarrival distribution[2]:

$$\begin{array}{lcl}
([a,b],[N_1,p_1],...,[N_i,p_i],...,[N_K,p_K]) & \rightarrow & ([a+1,b],[N_1,p_1],...,[N_i,p_i],...,[N_K,p_K]) \\
 & & 1\{a < r_b\}\lambda_a \\
([a,b],[N_1,p_1],...,[N_i,p_i],...,[N_K,p_K]) & \rightarrow & ([1,c],\delta([N_1,p_1]),...,\delta([N_1,p_i]),...,\delta([N_K,p_K])) \\
 & & 1\{a = r_b\}1\{\xi = c\}\lambda_a
\end{array}$$

where ξ is function of choosing a new series distribution out of totally R parallel distributions and

$$\delta([N_i,p_i]) = \begin{cases} [N_i+1,p_i] & \text{if } p_i \neq 0 \\ [0,1] & \text{if } p_i = 0 \end{cases}$$

If the event is a departure from the i^{th} server, the elements of the rate matrix have the following values.

$$\begin{array}{lcl}
([a,b],[N_1,p_1],...,[N_i,p_i],...,[N_K,p_K]) & \rightarrow & ([a,b],[N_1,p_1],...,[N_i,p_i+1],...,[N_K,p_K]) \\
 & & 1\{p_i < k\}k\mu \\
([a,b],[N_1,p_1],...,[N_i,p_i],...,[N_K,p_K]) & \rightarrow & ([a,b],[N_1,p_1],...,[N_i-1,1],...,[N_K,p_K]) \\
 & & 1\{p_i = k\}1\{N_i > 0\}k\mu \\
([a,b],[N_1,p_1],...,[N_i,p_i],...,[N_K,p_K]) & \rightarrow & ([a,b],[N_1,p_1],...,[0,0],...,[N_K,p_K]) \\
 & & 1\{p_i = k\}1\{N_i = 0\}k\mu
\end{array}$$

[1] If the arrival process is Poisson and task service times are exponential, the state space description $(N_1, \ldots, N_K)$ will be suffice where N_i is the number of tasks in the i-th server.

[2] We assume that the arrival process for stage $[a, b]$ has rate of λ_a and the mean arrival rate of external jobs is λ.

Given the probability steady state vector, the expected job response time can be computed by:

$$T = \frac{\text{Expected number of tasks in the system}}{\text{arrival rate of tasks}} = \frac{\sum_{s \in \mathcal{S}} \pi(s) f(s)}{K\lambda} \tag{4}$$

where $\mathcal{S}$ is the state space of the Markov model M, $\pi(s)$ is the steady state probability of state s and $f(s)$ expresses that total number of tasks in the system given the state is state s. The function $f(s)$ can be expressed as:

$$\begin{aligned} f(s) &= \sum_{i=1}^{K} \hat{N}_i + \sum_{i=1}^{K} (N^* - \hat{N}_i) \\ &= KN^* \end{aligned} \tag{5}$$

The above model does not possess a known closed form solution, and it is not possible to solve the problem numerically due to its infinite state space cardinality. The methodology we choose to solve this model is as follow. We first construct two models that can closely bound the performance (in our case, the performance measure is the expected job response time), then we show that these modified models can be evaluated *efficiently* by numerical methods.

Before we go on to the next section, let us describe briefly the intuition behind the construction of the models that provide the upper and lower bounds on the expected job response time. It is reasonable to argue that the distribution of stationary state probabilities for the model M are *skewed*, that is, the probability mass is concentrated in some relatively small subset of the state space, rather than distributed uniformly over the entire state space. The reasoning is that when job arrives into the system, it adds load uniformly to all K servers, therefore a job arrival will not increase the difference in the waiting queue length between different servers of the system. On the other hand, since the system is homogeneous, all servers are servicing their workload at the same rate, therefore, the probability of having a large difference in the waiting queue length among all servers should be relatively small. For example, for a 4-server system with Poisson arrivals and exponential service times, the steady state probability for state $(8,8,8,8)$ should have a higher state probability than state $(29,1,1,1)$. This insight suggests that we should, in any modified model, represent the exact behavior (transition rates) for the most *popular* states of the original model M. Note that the number of states in the most popular subset will be a function of the accuracy demanded and the computational cost one is willing to pay. When the modified system leaves these *popular* states, we modify the behavior of the system in such a manner that we can:

1. prove the modified model provides a bound (either upper or lower) for the original model and,

2. solve the modified system efficiently.

In the next section, we will describe the upper bound model and in the following section, the lower bound model will be presented.

3 Upper Bound Model M^u

In this section, we first describe the behavior of the upper bound model M^u, then we prove that M^u indeed provides an upper bound for the expected job response time for model M. We also develop a computational procedure for efficiently evaluating the expected job response time for model M^u. As in the original model, the upper bound model M^u has K parallel servers. A job arrives to the system according to a general arrival process with mean rate λ. Upon arrival, a job splits into K tasks (each having a k-stage Erlang distribution with mean $1/\mu$) and task t_i is assigned to the i^{th} server of the system. Each server will service tasks in a FCFS order. Again, a job departs from the system when its last task is completed.

In additional to the above description, the upper bound model M^u has two additional parameters, they are:

Definition 1 *Let d be a predefined threshold setting in the upper bound model M^u. We require that at any instant in time, the difference in the number of waiting foreground tasks*[3] *in any two queues is less than or equal to d.*

Remarks: In M^u, we force task departures to observe the following rule. When a task tries to depart from the service area, if its departure would cause the *difference* between the longest and shortest queue of waiting foreground tasks to exceed d, then this task is not allowed to depart but must remain in service and repeat its last stage of service. Note that we only need to check this condition at a task departure instant. Task arrival instants will never increase the difference in the waiting queue lengths. An important point is that since the job response time is a function of its last departing task, disallowing a task departure due to violation of the d threshold implies we are delaying the departure of a task that would have to wait in the synchronization area anyway (since the task is in one of the shortest queues). The delay of completion of a task can, intuitively, only delay the completion of some jobs and therefore, we suspect, should give a good upper bound on the mean job response time. We present the formal proof in the next subsection.

[3]definitions of foreground tasks and background tasks will be defined formally later.

With the restriction on the difference of the waiting queue length, then we have reduced the state space but it is still infinite. The next definition (and associated modification to the model) is used to impose a regular, repeating structure to the model which will then allow an effective solution procedure.

Definition 2 *Let $\overline{\mathbf{C}}$ be a K-dimensional vector, $\overline{\mathbf{C}} = (C, \ldots, C)$, where C is a predefined trigger threshold for the upper bound model M^u. If a job arrives and finds that there are already C waiting foreground tasks in any server, the system will suspend all tasks (except tasks from the current arriving job) and put them into background queues. A new busy cycle begins with the current arriving job.*

Remarks: A busy cycle starts when an arriving job causes the trigger threshold to be exceeded (or when the arrival finds the system is idle) and the newly arriving job starts the new busy cycle. The busy cycle ends when the foreground queues all all empty. When a busy cycle ends, the last suspended set of tasks (if any) are put into the foreground queues. Note that busy cycles are nested, that is during a busy cycle, an external job can arrive and trigger another busy cycle. When a busy cycle ends, only the set of tasks suspended at the initiation of that busy period is released for service. Therefore, a foreground task is defined to be any task which arrived during the currently active busy period. As mentioned above, the purpose of the trigger threshold $\overline{\mathbf{C}}$ is to create a transition structure that yields efficient numerical solution.

3.1 Proof that M^i provides an upper bound on expected job response time

In this section, we will prove that the model M^u provides an upper bound on the expected job response time[4]. The approach is to show that M^u gives an upper bound on the number of tasks in the system at any given point in time using sample path analysis [19]. In the case that the model exhibits stationary behavior, we apply Equation (4) to obtain the expected job response time.

To establish the upper bound, let us define the notion of *sub-task*. Since each task has a service time which is a random variable with a k-stage Erlang distribution, whenever a task arrives to a server, it generates k sub-tasks to that server. To prove the upper bound, let us concentrate on the time instants when certain events occur. There are two kinds of events of interest, namely, job arrivals and sub-task departures. In the latter case, it

[4]The proof for the bounding process is applicable for a general arrival process. In a later section, we will define the class of arrival processes we can handle with our computational algorithm.

is useful to think of each server as continuously serving customers. Hence a *service event* at server k occurs as a Poisson process with parameter $k\mu$ (note that a service event is also a departure event only when there is a sub-task in the server). For the original model M, let us define $\boldsymbol{\mathcal{N}}_i = (\mathcal{N}_{i,1}, \cdots, \mathcal{N}_{i,K})$ to be the joint queue lengths of sub-tasks immediately after the i^{th} event. Let $\boldsymbol{\mathcal{N}}_0$ denote the initial queue lengths. We have the following evolution equations. If the $(i+1)^{th}$ event is a job arrival, then,

$$\mathcal{N}_{i+1,l} = \mathcal{N}_{i,l} + k \qquad 1 \leq l \leq K \tag{6}$$

If the $(i+1)^{th}$ event is a sub-task service event at the j^{th} server, then the joint queue length will be:

$$\mathcal{N}_{i+1,l} = \begin{cases} \mathcal{N}_{i,l}, & l \neq j, \\ (N_{i,j} - 1)^+, & l = j. \end{cases} \tag{7}$$

For the upper bound model M^u, we define a binary valued random variable Y_i that takes on the value 0 if a sub-task is not allowed to depart and the value 1 if a sub-task is allowed to depart at the i^{th} event (provided that it is a service event). Note that the random variable Y_i for the original model M is always equal to 1 while for the upper bound model, Y_i can be 0 or 1 depending on the situation. Specifically, Y_i will take on value of 0 when:

1. the sub-task departure would violate the threshold constraint d and,
2. departure of background sub-tasks (or equivalently, tasks) is not allowed due to unfinished foreground sub-tasks.

Let $\boldsymbol{\mathcal{N}}^u(t)$ be the joint queue lengths for all types of sub-tasks (active and background) in the upper bound model M^u. We have the following evolution equations at the time of arrival and service events. If the $(i+1)^{th}$ event corresponds to an job arrival,

$$\mathcal{N}^u_{i+1,l} = \mathcal{N}^u_{i,l} + k \qquad 1 \leq l \leq K \tag{8}$$

If it is a task service event at server j,

$$\mathcal{N}^u_{i+1,l} = \begin{cases} \mathcal{N}^u_{i,l}, & l \neq j, \\ (\mathcal{N}^u_{i,j} - Y_{i+1})^+, & l = j. \end{cases} \tag{9}$$

Definition 3 *Let $\boldsymbol{X}$ and $\boldsymbol{Y}$ be a vector of real values of random variables. We say $\boldsymbol{Y}$ is stochastically larger than $\boldsymbol{X}$, written as $\boldsymbol{X} \leq_{st} \boldsymbol{Y}$ iff*

$$E[h(\boldsymbol{X})] \leq E[h(\boldsymbol{Y})]$$

for all increasing functions h.

Theorem 1 *If $\boldsymbol{\mathcal{N}}(0) \leq_{st} \boldsymbol{\mathcal{N}}^u(0)$, then $\boldsymbol{\mathcal{N}}(t) \leq_{st} \boldsymbol{\mathcal{N}}^u(t)$.*

Proof: Couple the initial joint queue length of sub-tasks such that $\mathcal{N}(0) \leq \mathcal{N}^u(0)$. Condition on the initial joint queue length, arrival event instants and service event instants. The proof is by induction on all event times to establish the deterministic relationship:

$$\boldsymbol{\mathcal{N}}_i \leq \boldsymbol{\mathcal{N}}_i^u, \quad i \geq 0.$$

where $\leq$ is taken to mean componentwise.

For $i = 0$, $\boldsymbol{\mathcal{N}}(0) \leq \boldsymbol{\mathcal{N}}^u(0)$.

For the induction step, assume that $\boldsymbol{\mathcal{N}}_i \leq \boldsymbol{\mathcal{N}}_i^u$ holds for $i = k$. For $i = k+1$, if the i^{th} event is a job arrival, the inequality obviously holds since both models never reject any task arrival. If the i^{th} event is a service event, since Y_i is less than or equal to 1 for model M^u, the relationship holds also. Therefore we have $\boldsymbol{\mathcal{N}}(t) \leq \boldsymbol{\mathcal{N}}^u(t)$. By removing the conditioning on initial joint queue length of sub-tasks, arrival event times and service event times, we have:

$$\boldsymbol{\mathcal{N}}(t) \quad \leq_{st} \quad \boldsymbol{\mathcal{N}}^u(t), \qquad for \quad t \geq 0$$

∎

Corollary 1 *If $\boldsymbol{\mathcal{N}}(0) \leq_{st} \boldsymbol{\mathcal{N}}^u(0)$, then $T \leq T^u$ where T (T^u) is the expected job response time for model M (M^u).*

Proof: Let $\boldsymbol{N}(t)$ and $\boldsymbol{N}^u(t)$ be the joint queue lengths in model M and M^u at time t. We have:

$$\begin{aligned}\boldsymbol{N}(t) &= (\lceil \mathcal{N}_1(t)/k \rceil, \ldots, \lceil \mathcal{N}_K(t)/k \rceil) \\ \boldsymbol{N}^u(t) &= (\lceil \mathcal{N}_1^u(t)/k \rceil, \ldots, \lceil \mathcal{N}_K^u(t)/k \rceil)\end{aligned}$$

Based on Theorem 1, we have $\boldsymbol{\mathcal{N}}(t) \leq_{st} \boldsymbol{\mathcal{N}}^u(t)$. Since $\lceil \mathcal{N}_i(t)/k \rceil \leq \lceil \mathcal{N}_i^u(t)/k \rceil$, we have $\boldsymbol{N}(t) \leq_{st} \boldsymbol{N}^u(t)$. Now, let $\boldsymbol{N} = \lim_{t \to \infty} \boldsymbol{N}(t)$ and $\boldsymbol{N}^u = \lim_{t \to \infty} \boldsymbol{N}^u(t)$. Since the function f define in Equation (5) is an increasing function, then based on Equation (4), we have:

$$T \leq T^u$$

∎

3.2 Computational procedure for upper bound model M^u

In this section, we discuss the computational procedure to obtain the expected response time for the upper bound model M^u. Note that in the proof that M^u does yield an upper bound for the mean response time, the interarrival time distribution of jobs can be general. In this section, the class of interarrival process that we can handle in our computational algorithm for the upper bound model M^u is more restrictive.

Let τ denote the random variable equal to the interarrival time distribution of a job to the fork-join queueing system and $A^*(s)$ denotes its Laplace Transform. The class of interarrival distributions we allow has the form:

$$A^*(s) \;=\; \sum_{j=1}^{R} \alpha_j \prod_{i=1}^{r_j} \frac{\lambda_{ij}}{s+\lambda_{ij}} \qquad \text{for } 0 < \alpha_j < 1, \sum_{j=1}^{R} \alpha_j = 1 \text{ and } \lambda_{ij} > 0 \tag{10}$$

such that $-\frac{dA^*(s)}{ds}|_{s=0} = \lambda^{-1}$. This class of series-parallel distributions includes such common distributions like exponential, Erlang, hyperexponential, ..., etc. Figure 2 illustrates this class of series-parallel distribution where each stage represents an exponentially distributed random variable τ_{ij} with parameter λ_{ij}.

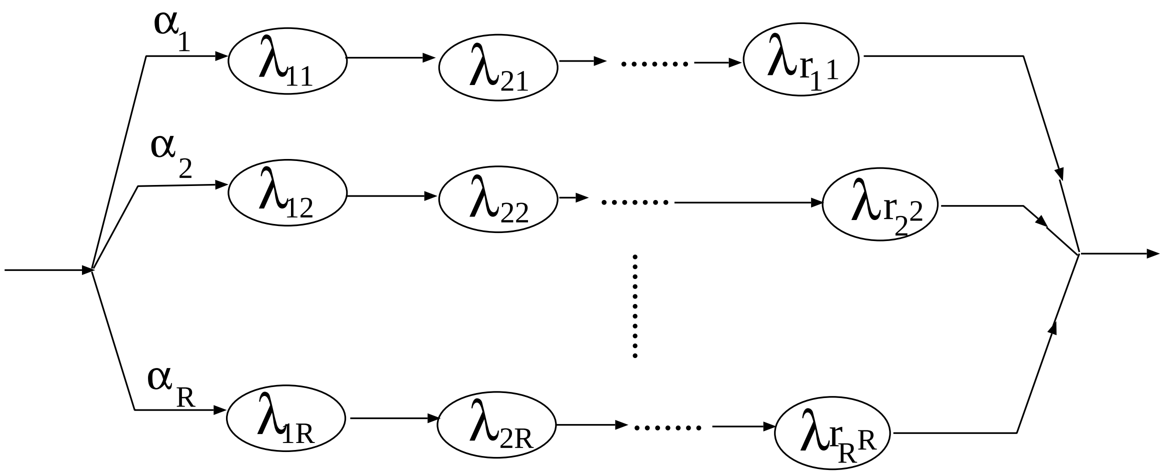

Figure 2: Series-parallel distribution for the interarrival time distribution in M^u.

To describe the computational procedure for the upper bound model M^u, it is important that we first describe the state space of the upper bound model. First, let us define the following:

Definition 4 *Given parameters K, d, $\overline{\mathbf{C}}$, external job interarrival process and the tasks service time distribution, let $\mathcal{S}_{tt}$ be an ordered list of states for foreground tasks in the* *upper bound model M^u such that any additional job arrival will cause the system to switch to a new busy cycle.*

Example 1: If the arrival process is Poisson, and task service time is exponentially distributed, $K = 3$, $d = 2$, $\overline{\mathbf{C}} = (5,5,5)$, then

$$\begin{aligned}\mathcal{S}_{tt} \;=\; &\{(3,3,5),(3,4,5),(3,5,3),(3,5,4),(3,5,5),(4,3,5),\\ &(4,4,5),(4,5,3),(4,5,4),(4,5,5),(5,3,3),(5,3,4),\\ &(5,3,5),(5,4,3),(5,4,4),(5,4,5),(5,5,3),(5,5,4),(5,5,5)\}\end{aligned}$$

Example 2: If the job arrival process and the task service times are two-stage Erlang, $K = 2$, $d = 2$, $\overline{\mathbf{C}} = (3,3)$, then

$$\begin{aligned}\mathcal{S}_{tt} \;=\; &\{(1,[1,1],[3,1]),(1,[1,1],[3,2]),(2,[1,2],[3,1]),(2,[1,2],[3,2]),\\ &(1,[1,2],[3,1]),(1,[1,2],[3,2]),(2,[1,2],[3,1]),(2,[1,2],[3,2]),\\ &(1,[2,1],[3,1]),(1,[2,1],[3,2]),(2,[2,1],[3,1]),(2,[2,1],[3,2]),\\ &(1,[2,2],[3,1]),(1,[2,2],[3,2]),(2,[2,2],[3,1]),(2,[2,2],[3,2]),\\ &(1,[3,1],[1,1]),(1,[3,1],[1,2]),(2,[3,1],[1,1]),(2,[3,1],[1,2]),\\ &(1,[3,1],[2,1]),(1,[3,1],[2,2]),(2,[3,1],[2,1]),(2,[3,1],[2,2]),\\ &(1,[3,1],[3,1]),(1,[3,1],[3,2]),(2,[3,1],[3,1]),(2,[3,1],[3,2]),\\ &(1,[3,2],[1,1]),(1,[3,2],[1,2]),(2,[3,2],[1,1]),(2,[3,2],[1,2]),\\ &(1,[3,2],[2,1]),(1,[3,2],[2,2]),(2,[3,2],[2,1]),(2,[3,2],[2,2]),\\ &(1,[3,2],[3,1]),(1,[3,2],[3,2]),(2,[3,2],[3,1]),(2,[3,2],[3,2])\}\end{aligned}$$

Definition 5 *Let B be the cardinality of the set of trigger states $\mathcal{S}_{tt}$, or $B = |\mathcal{S}_{tt}|$.*

Definition 6 *Let $\mathcal{S}_{tt}(i)$ be the i^{th} element in $\mathcal{S}_{tt}$, where $0 \le i \le B-1$.*

Using example 1 listed above, we have:

$$B \;=\; 19; \qquad \mathcal{S}_{tt}(0) = (3,3,5),\ldots,\mathcal{S}_{tt}(18) = (5,5,5)$$

Using example 2 listed above, we have:

$$B \;=\; 40; \qquad \mathcal{S}_{tt}(0) = (1,[1,1],[3,1]),\ldots,\mathcal{S}_{tt}(39) = (2,[3,2],[3,2])$$

Definition 7 *Let $S^u_{i,j}$, $i \ge 0, 0 \le j \le B^i - 1$, be the set of states for the upper bound model M^u such that (1) system is in the i^{th} (where $i > 0$) nested level of busy cycles,* *(2) the difference in the waiting queue length between foreground tasks among different servers is less than or equal to d and, (3) the system entered the i^{th} nested busy cycle from the $(i-1)^{th}$ busy cycle through the trigger state $\mathcal{S}_{tt}(j \bmod B) \subset S^u_{i-1,j/B}$, where "/" is an integer divide operator.*

Transitions between states in $S^u_{i,j}$ behave like the original model M except :

- if a task departure from the server would cause the difference between the longest and shortest queue of waiting foreground to exceed d, then this task is not allowed to depart but must remain in service and repeat its last stage of service.
- If a new job arrives and the system is in one of the trigger states, for example $\mathcal{S}_{tt}(l)$, then a new busy cycle begins with this newly arrive job and the system enters state $(1, [0,k], \cdots, [0,k])$ in $S^u_{i+1,jB+l}$.
- Upon completion of the last foreground job in $S^u_{i,j}$, the previously suspended set of tasks is released for service. That is, the system will make a transition back to one of the trigger states in $S^u_{i-1,j/B}$.

With this definition, we can see that the state space of the upper bound model M^u can be organized as a tree such that each node of the tree represents the set of states $S^u_{i,j}$ of all possible waiting queue lengths for active tasks. Each state in the model has four components, namely (1) a stack of *return states*[5] $(s_1, \ldots, s_i)$ where i is the depth of the node in the tree and, (2) the stage of the interarrival distribution, (3) the waiting queue lengths of active tasks at each servers and (4) the stage of service for each task in service. All states in the same node of the tree have the same first component, i.e, the same stack of return states. Figure 3 illustrates the case where the arrival process is Poisson and task service times are exponentially distributed, $K = 2, d = 2, C = 5$, $\mathcal{S}_{tt} = \{(3,5), (4,5), (5,3), (5,4), (5,5)\}$.

As stated above, upon departure of the last foreground job in $S^u_{i,j}$, there will be a transition back to one of the trigger states in $S^u_{i-1,j/B}$. At this point, we change the arrival process so that we can have an efficient algorithm for solving the upper bound model M^u. First, we define random variables $\hat{\tau}_{kj}$ based on the notation used in Equation (10) :

$$\hat{\tau}_{kj} \;=\; \sum_{l=k}^{r_j} \tau_{lj} \qquad \text{for } 1 \le j \le R \text{ and } 1 \le k \le r_j \tag{11}$$

[5]In the actual state representation, we do not need to represent the stack of return states since they can be derived from the relative position within the tree.

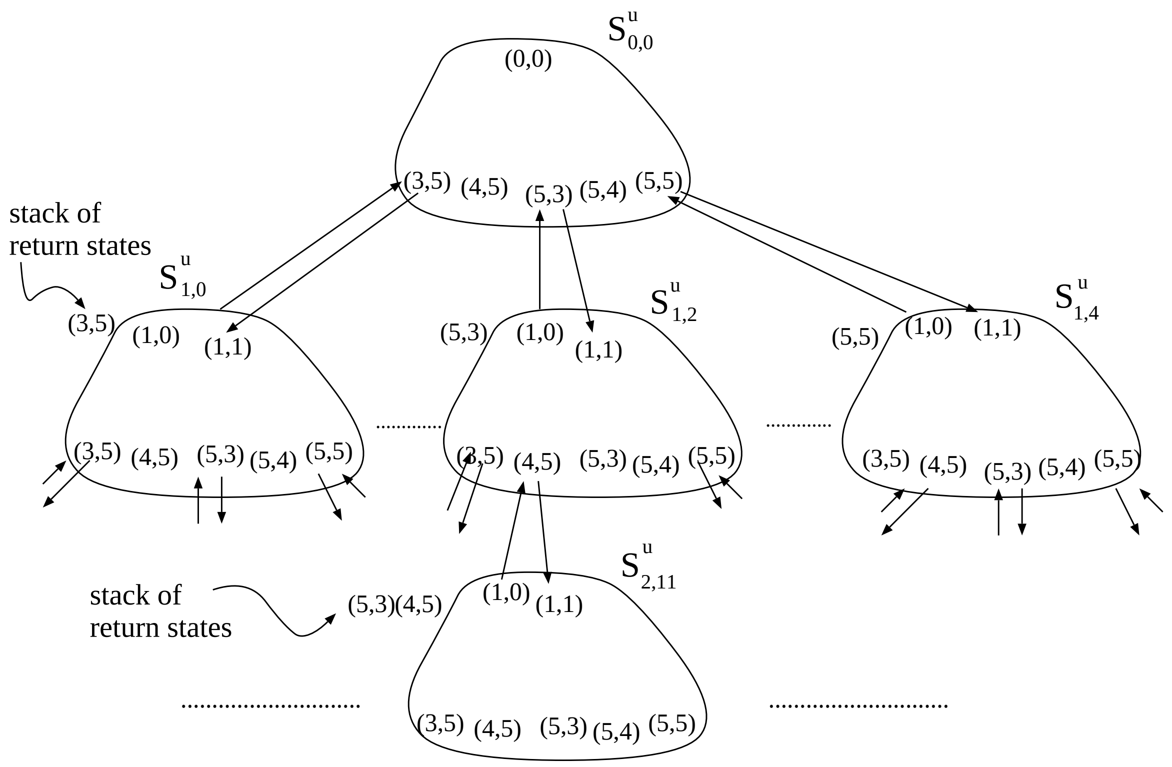

Figure 3: State Space Partition for Upper Bound Model M^u.

with the corresponding Laplace transform:

$$\hat{A}^*_{kj}(s) = \prod_{l=k}^{r_j} \left(\frac{\lambda_{lj}}{s + \lambda_{lj}} \right)$$

We can now express the Laplace transform of the job interarrival time τ as:

$$A^*(s) = \sum_{j=1}^{R} \alpha_j \hat{A}^*_{1j}(s)$$

We also define τ_{max} to be an exponentially distributed random variable with parameter λ_{max} such that:

$$\lambda_{max} = \max_{1 \le j \le R} \lambda_{r_j j}$$

The Laplace transform of τ_{max} is $A^*_{\lambda_{max}}(s) = (\frac{\lambda_{max}}{s+\lambda_{max}})$. The following theorem relates τ and τ_{max}.

Theorem 2 $\tau_{max} \leq_{st} \tau$.

Proof: First we show $\tau_{max} \leq_{st} \hat{\tau}_{kj}$ for $1 \leq j \leq R$ and $1 \leq k \leq r_j$. Since

$$\begin{aligned} \overline{F}_{\tau_{max}}(x) &= e^{-\lambda_{max} x} && \text{for } x \geq 0 \text{ and} \\ \overline{F}_{\tau_{r_j,j}}(x) &= e^{-\lambda_{r_j,j} x} && \text{for } x \geq 0, \end{aligned}$$

we have $\overline{F}_{\tau_{max}}(x) \leq \overline{F}_{\tau_{r_j,j}}(x)$ for all $x \geq 0$, therefore $\tau_{max} \leq_{st} \tau_{r_j,j}$. Since

$$\hat{\tau}_{k,j} = \tau_{k,j} + \cdots + \tau_{r_j,j} \qquad \text{for } 1 \leq j \leq R \text{ and } 1 \leq k \leq r_j$$

and τ_{ij} are non-negative random variables, it is easy to see that $\tau_{max} \leq_{st} \hat{\tau}_{k,j}$ for $1 \leq j \leq R$ and $1 \leq k \leq r_j$. The interarrival time random variable τ, can be expressed as

$$\tau = \sum_{j=1}^{R} \alpha_j \hat{\tau}_{1j}$$

Because $\leq_{st}$ relationship is preserved under linear combination [28]. Therefore:

$$\tau_{max} = \sum_{j=1}^{R} \alpha_j \tau_{max} \quad \leq_{st} \quad \sum_{j=1}^{R} \alpha_j \hat{\tau}_{1j} = \tau \tag{12}$$

∎

Let Z be a random variable that takes on value 0 or 1. Define

$$\hat{\tau}_{max} = \begin{cases} \tau & \text{Prob}(Z=0) \\ \tau_{max} & \text{Prob}(Z=1) \end{cases}$$

Corollary 2 $\hat{\tau}_{max} \leq_{st} \tau$.

Proof: Since

$$\text{Prob}(Z=1)\tau + \text{Prob}(Z=0)\tau_{max} \leq_{st} \text{Prob}(Z=1)\tau + \text{Prob}(Z=0)\tau$$

therefore,

$$\hat{\tau}_{max} \leq_{st} \tau$$

∎

Let $n_F = \{n_F(t), t \geq 0\}$ and let $n_G = \{n_G(t), t \geq 0\}$ denote the stochastic processes defined as the number of tasks in the queueing system with interarrival distribution $F_{\hat{\tau}_{max}}$ and G_τ, respectively. We want to show that

$$n_G(t) \leq_{st} n_F(t) \qquad \text{for } t \geq 0$$

In preparation for this proof, we need the following lemma about *coupling* between random variables [28]:

Lemma 1 *If F and G are probability distributions such that $\overline{F}(a) \leq \overline{G}(a)$ for all a, then there exist random variables X and Y having distributions F and G respectively such that*

$$P\{Y \geq X\} = 1$$

Theorem 3 $\{n_G(t), t \geq 0\} \leq_{st} \{n_F(t), t \geq 0\}$.

Proof: To prove the above statement, we constrain the system such that (1) the service times of tasks are exponential and, (2) there is no a prior knowledge of service time for the scheduling discipline. Now only observe the process when there is an arrival. We use the *coupling* argument as follows. Let $X_1, X_2, \ldots$ be independent and distributed according to G_τ. Then the process generated by X_i, called it n_G^*, has the same distribution as n_G. Now generate independent random variables $Y_1, Y_2, \ldots$ having distribution of $F_{\hat{\tau}_{max}}$ such that $Y_i \leq X_i$. Then the process generated by interarrival time Y_i, called it n_F^*, has the same probability distribution as n_F. Since $Y_i \leq X_i$ for all i, it follows that

$$\begin{aligned} n_F^*(t) &\leq n_G^*(t) && \text{for all } t \\ n_F(t) &\leq n_G(t) && \text{for all } t \text{ and} \\ n_F &\leq_{st} n_G && \end{aligned} \tag{13}$$

∎

The significance of Theorem 2, Corollary 2 and Theorem 3 is as follow. At some *imbedded times*, we can modify the interarrival distribution according to Theorem 2 such that we have an upper bound on the number of tasks in the system. These imbedded times are chosen to be the times at which all the foreground tasks in the model are finished and the system switches to the last suspended set of tasks, which are now to be the foreground tasks. Therefore, we have the interarrival process as specified in Equation (10) except when a suspended set of tasks are activated. When this occurs, the next arrival occurs is an exponential time distribution with rate λ_{max}. Following this arrival, the original interarrival process is again in effect.

Due to the routing of task arrivals, the constraint on tasks departures and the modification of interarrival time distribution as indicated in Corollary 2 at points where the system switches from $S^u_{i,j}$ to $S^u_{i-1,k}$, we can show that the only transition from set $S^u_{i-1,k}$ to set $S^u_{i,j}$ is actually to one state in $S^u_{i,j}$. Further, transitions from $S^u_{i+1,k}$ to $S^u_{i,j}$ can only go to one state in $S^u_{i,j}$. This unique transition structure, as we will show later, allows us to efficiently solve the model via *exact* aggregation [7]. After we aggregate each set of states, $S^u_{i,j}$, into a single state $s^u_{i,j}$, the aggregated model has a tree structure, which implies the aggregated process is time reversible [15] and the steady state probability solution to this aggregated process can be derived easily.

We are now in a position to describe the computational procedure to obtain the expected response time for the upper bound model M^u. Let $Q_{S^u_{i,j},S^u_{l,m}}$ be the transition rate matrix from states in $S^u_{i,j}$ to states in $S^u_{l,m}$. Based on the tree structure of the transitions between states as depicted in Figure 3, a brief description of how to obtain T^u is as follow:

1. compute the conditional state probability given that the system is in set $S^u_{i,j}$.
2. aggregate each $S^u_{i,j}$ into a single state $s^u_{i,j}$
3. compute the transition rates between aggregate states.
4. given the conditional state probabilities of $S^u_{i,j}$ and aggregate state probabilities $s^u_{i,j}$, by applying Equation (4) and Equation (5), we can find the T^u, the expected job response time for the upper bound model M^u.

Remarks: In computing the conditional probabilities given the system is in $S^u_{i,j}$, observe that the transition from its parents $S^u_{i-1,k}$ is to one state in $S^u_{i,j}$ and the transitions from each of its children $S^u_{i+1,l}$ are through exactly one state in $S^u_{i,j}$ respectively.

In order to justify the procedure for computing the conditional steady state probabilities for states in $S^u_{i,j}$ given that the system is in set $S^u_{i,j}$, we need the follow lemma from [8, 23, 20]:

Lemma 2 *Given an irreducible Markov process with state space $S = A \cup B$ and transition rate matrix:*

$$\begin{bmatrix} Q_{A,A} & Q_{A,B} \\ Q_{B,A} & Q_{B,B} \end{bmatrix}$$

where $Q_{i,j}$ is the transition rate sub-matrix from partition i to partition j. If $Q_{B,A}$ has all zero entries except for some non-zero entries in the i-th column, the conditional steady *state probability vector, given that the system is in partition A, is the solution to the following system of linear equations:*

$$\begin{aligned}\vec{\pi}_{|A}\left[Q_{A,A} + Q_{A,B}\ \underline{e}\ \underline{e}_i^T\right] &= \vec{0}\\ \vec{\pi}_{|A}\ \underline{e} &= 1\end{aligned}$$

where $\underline{e}_i^T$ is a row vector with a 0 in each component, except for the i-th component which has the value 1.

The implication of the above lemma is that there is one state in A by which A is entered (from B), then we can compute the conditional state probability of system in A without knowing the transition structure,$Q_{B,B}$. The following theorem shows that how we can apply Lemma 2 to compute the conditional state probability vector given system is in $S^u_{i,j}$.

Theorem 4 *Let $\mathcal{M}$ be a Markov process with rate matrix $\hat{Q}_{S^u_{i,j},S^u_{i,j}}$, which is equal to $Q_{S^u_{i,j},S^u_{i,j}}$ except for the following modifications:*

1. *A transition to the parent set of states $S^u_{i-1,k}$ is changed to a transition to the initial state of $S^u_{i,k}$ (which is the state where $S^u_{i,k}$ is entered from $S^u_{i-1,k}$.)*
2. *Each transitions to a child in set $S^u_{i+1,l}$ is changed to a transition to state s, where state s is the transition from $S^u_{i+1,l}$ to $S^u_{i,j}$.*

The steady state probability vector for the Markov process with rate matrix $\hat{Q}_{S^u_{i,j},S^u_{i,j}}$ is the conditional state probability vector given the system is in set $S^u_{i,j}$.

Proof: Without loss of generality, we show how to compute the conditional state probabilities given that the system is in $S^u_{1,0}$. Let us partition the rate matrix of the model M^u as follow:

$$\begin{bmatrix} Q_{S^u_{1,0},S^u_{1,0}} & Q_{S^u_{1,0},T_1} & \cdots & Q_{S^u_{1,0},T_B}, & Q_{S^u_{1,0},S'_T} \\ Q_{T_1,S^u_{1,0}} & Q_{T_1,T_1} & 0 & 0 & 0 \\ \vdots & 0 & \ddots & 0 & 0 \\ Q_{T_B,S^u_{1,0}} & 0 & \vdots & Q_{T_B,T_B} & 0 \\ Q_{S'_T,S^u_{1,0}} & 0 & 0 & 0 & Q_{S'_T,S'_T} \end{bmatrix}$$

where $T_i, 1 \le i \le B$ is the i^{th} sub-tree under $S^u_{1,0}$ and T' is :

$$T' = \mathcal{S}^u - S^u_{1,0} - \bigcup_{i=1}^{B} T_i$$

and $\mathcal{S}^u$ is the state space of the upper bound model M^u. Note that transition matrices $Q_{T',S^u_{1,0}}$ and $Q_{T_i,S^u_{1,0}}$, for $i \geq 1$ are transition rate matrices with one non-zero column vector, this implies that there is only one way to enter $S^u_{1,0}$ through its parent and only one way via its children. By Lemma 2, the modification (1) provides the conditional state probabilities vector given the system is in $S^u_{1,1}$ and all its children. Now applying Lemma 2 repeatedly for each subtree of $S^u_{1,0}$, we obtain the conditional state probabilities given that the system is in $S^u_{1,0}$ ∎

Due to the way we define our busy cycles, transition rate matrices $\hat{Q}_{S^u_{i,j},S^u_{i,j}}$ are the same for $i \geq 1$. This implies we only need to solve two models, $\hat{Q}_{S^u_{0,0},S^u_{0,0}}$ and $\hat{Q}_{S^u_{1,0},S^u_{1,0}}$, to obtain all the conditional state probabilities. Once we obtain the conditional state probabilities for partition $S^u_{i,j}$, we can aggregate each partition [7] into a single state, which we denote as $s^u_{i,j}$. After aggregation, the state transition structure forms a tree as depicted in Figure 4 where $\lambda^{u'}_i$, $0 \leq i \leq B-1$ can be derived from the conditional steady

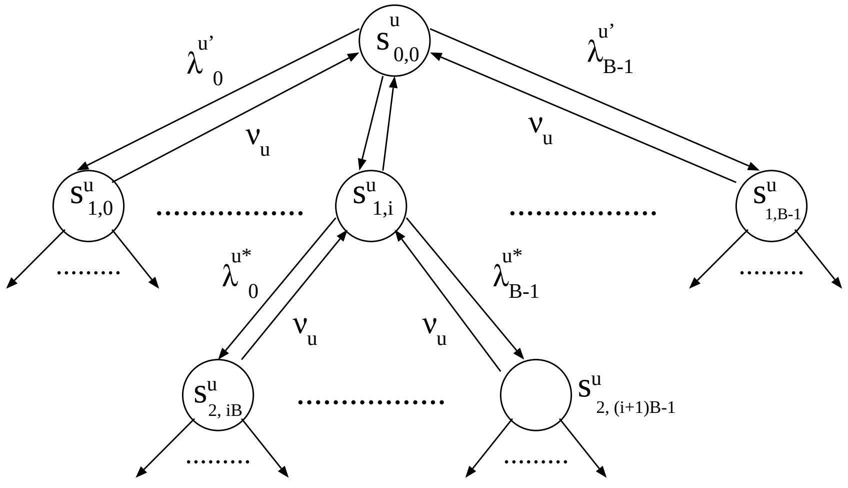

Figure 4: Aggregated Process for upper bound model M^u.

state probabilities of $\hat{Q}_{S^u_{0,0},S^u_{0,0}}$ and $\lambda^{u*}_i, 0 \leq i \leq B-1$ can be derived from the conditional steady state probabilities of $\hat{Q}_{S^u_{1,0},S^u_{1,0}}$. Let Ψ be the set of states in $S^u_{i,j}$, $i \geq 1$, such that there are transitions from Ψ back to the parent of $S^u_{i,j}$. Then we have:

$$\lambda^{u'}_i = \hat{\pi}'(a)\frac{1}{\tau_{r_j j}} \qquad \text{for } a \in \mathcal{S}_{tt} \text{ where } \mathcal{S}_{tt} \subset S^u_{0,0},\ 0 \leq i \leq B-1,\ 1 \leq j \leq R \quad (14)$$

$$\lambda^{u*}_i = \hat{\pi}(a)\frac{1}{\tau_{r_j j}} \qquad \text{for } a \in \mathcal{S}_{tt} \text{ where } \mathcal{S}_{tt} \subset S^u_{1,0},\ 0 \leq i \leq B-1,\ 1 \leq j \leq R \quad (15)$$

$$\nu_u = \sum_{s \in \Psi} \hat{\pi}(s) k \mu \quad (16)$$

where $\hat{\pi}'(a)$ and $\hat{\pi}(a)$ are the steady state probability for Markov processes with rate matrix $\hat{Q}_{S^u_{0,0},S^u_{0,0}}$ and $\hat{Q}_{S^u_{1,0},S^u_{1,0}}$ respectively.

Let $\pi^*(i,j)$ be the steady state probability for the aggregate state $s^u_{i,j}$. For this aggregated process, we can write the *local balance equations* [16]:

$$\begin{aligned}
\pi^*(1,i) &= \pi^*(0,0)\frac{\lambda^{u'}_i}{\nu_u} && 0 \le i \le B-1 \\
\pi^*(k,j) &= \pi^*(k-1,j/B)\frac{\lambda^{u*}_{(j \bmod B)}}{\nu_u} && k \ge 1, 0 \le j \le B^k - 1
\end{aligned}$$

Normalizing all local balance equations, we have:

$$\pi^*(0,0)\left[1+\frac{1}{\nu_u}\sum_{i=0}^{B-1}\lambda^{u'}_i+\left(\frac{1}{\nu_u}\right)^2\sum_{i=0}^{B-1}\sum_{j=0}^{B-1}\lambda^{u'}_i\lambda^{u*}_j+\left(\frac{1}{\nu_u}\right)^3\sum_{i=0}^{B-1}\sum_{j=0}^{B-1}\sum_{k=0}^{B-1}\lambda^{u'}_i\lambda^{u*}_j\lambda^{u*}_k+\cdots\right]=1$$

Define $\lambda^{u'} = \sum_{i=0}^{B-1}\lambda^{u'}_i$ and $\lambda^{u*} = \sum_{i=0}^{B-1}\lambda^{u*}_i$, we have:

$$\begin{aligned}
\pi^*(0,0) &= \left[1+\left(\frac{\lambda^{u'}}{\nu_u}\right)+\left(\frac{\lambda^{u'}}{\nu_u}\right)\frac{\lambda^{u*}}{\nu_u}+\left(\frac{\lambda^{u'}}{\nu_u}\right)\left(\frac{\lambda^{u*}}{\nu_u}\right)^2+\cdots\right]^{-1} \\
\pi^*(0,0) &= \left[1+\frac{\lambda^{u'}}{\nu_u}\sum_{i=0}^{\infty}\left(\frac{\lambda^{u*}}{\nu_u}\right)^i\right]^{-1} \\
\pi^*(0,0) &= \frac{\nu_u-\lambda^{u*}}{\nu_u+\lambda^{u'}-\lambda^{u*}} \qquad (17)
\end{aligned}$$

Let $\pi^*(i)$ be the steady state probabilities for all nodes at the i^{th} level of the aggregated process. $\pi^*(i)$ can be expressed as:

$$\pi^*(i) = \sum_{j=0}^{B^i-1}\pi^*(i,j) = \frac{\lambda^{u'}}{\nu_u}\left(\frac{\lambda^{u*}}{\nu_u}\right)^i\left(\frac{\nu_u-\lambda^{u*}}{\nu_u+\lambda^{u'}-\lambda^{u*}}\right) \qquad i \ge 1 \quad (18)$$

To compute the job response time T^u for the upper bound model, which is based on Equation (4), let us define $r^u_{f_0}$ to be the expected number of tasks given the system is in $S^u_{0,0}$, r^u_f to be the expected number of foreground tasks[6] given the system is in $S^u_{i,j}$ for $i \ge 1$ and $r^u_b(i)$ to be the expected number of background tasks given the system is in $S^u_{i,j}$ for $i \ge 1$. Therefore, given the rate matrix $\hat{Q}_{S^u_{0,0},S^u_{0,0}}$ and $\hat{Q}_{S^u_{i,j},S^u_{i,j}}$, we can compute $r^u_{f_0}$ and r^u_f. The number of background tasks (which includes the completed tasks associated

[6]Note that r^u_f is the same for $S^u_{i,j}$ for $i \ge 1$.

with the jobs that are placed in the background) given that the system is in i^{th} level of busy cycle is equal to $iK(C+1)$. Therefore, the total expected number of tasks in the upper bound model M^u is:

$$
\begin{aligned}
N^u &= \pi^*(0,0)r^u_{f_0} + \sum_{i=1}^{\infty} \frac{\lambda^{u'}}{\nu_u} \left(\frac{\lambda^{u*}}{\nu_u} \right)^{i-1} \pi^*(0,0) \left[iK(C+1) + r^u_f \right] \\
&= \pi^*(0,0)r^u_{f_0} + \frac{\lambda^{u'}}{\nu_u}\pi^*(0,0)r^u_f \sum_{i=0}^{\infty} \left(\frac{\lambda^{u*}}{\nu_u} \right)^{i} + \frac{\lambda^{u'}}{\nu_u}\pi^*(0,0)K(C+1) \sum_{i=0}^{\infty} \left(\frac{\lambda^{u*}}{\nu_u} \right)^{i} [i+1] \\
&= \pi^*(0,0) \left[r^u_{f_0} + \frac{r^u_f \lambda^{u'}}{\nu_u - \lambda^{u*}} + \frac{\lambda^{u'} K(C+1)\nu_u}{(\nu_u - \lambda^{u*})^2} \right] \\
&= \left(\frac{\nu_u - \lambda^{u*}}{\nu_u + \lambda^{u'} - \lambda^{u*}} \right) \left[r^u_{f_0} + \frac{r^u_f \lambda^{u'}}{\nu_u - \lambda^{u*}} + \frac{\lambda^{u'} K(C+1)\nu_u}{(\nu_u - \lambda^{u*})^2} \right] \qquad (19)
\end{aligned}
$$

The expected job response time for upper bound model M^u is:

$$
T^u = \left[\frac{\nu_u - \lambda^{u*}}{K\lambda\left(\nu_u + \lambda^{u'} - \lambda^{u*}\right)} \right] \left[r^u_{f_0} + \frac{r^u_f \lambda^{u'}}{\nu_u - \lambda^{u*}} + \frac{\lambda^{u'} K(C+1)\nu_u}{(\nu_u - \lambda^{u*})^2} \right] \qquad (20)
$$

4 Lower Bound Model

In this section, we first describe the behavior of the lower bound model M^l, then we prove that M^l indeed provides a lower bound for the expected job response time for the model M. We also develop a computational procedure for efficiently calculating the expected job response time for model M^l.

The lower bound model M^l is modeled as having K parallel servers. A job arrives to the system according to a general arrival process[7] with mean rate λ. Upon arrival, a job splits into K tasks and task t_i, $1 \leq i \leq K$, has a k-stage Erlang distribution with mean $1/\mu$, and is assigned to the i^{th} server of the system. Each server will service its assigned tasks in a FCFS order. Again, a job departs from the system when its last task is completed. Similar to the upper bound model M^u, we have a threshold d, which places a constraint on the maximum difference in foreground waiting queue lengths. There is also a threshold trigger $\overline{\mathbf{C}}$, such that if a job arrives and finds that there are already C foreground waiting tasks in any server, a new busy cycle begins with this newly arrived job. The busy cycle ends when all servers complete all tasks which arrive in the current busy cycle. Again, the definition of a busy cycle implies nested busy cycles, i.e., during

[7]In later section, we will define the class of arrival processes we can handle in the computational algorithm for the lower bound model M^l. The proof for the bounding process, however, is applicable for general arrival process.

a busy cycle, a job arrival can trigger the start of a new busy cycle. When a busy cycle ends, only the set of tasks suspended at the initiation of that busy cycle is released for service.

The lower bound model operates as follows:

- If a task departure would violate the threshold d, it remains in the server for another phase of service (e.g: for k-stage Erlang, the task remains in the last stage of service) and we force a foreground task departure from one of the longest queues.
- When the system has some background tasks waiting and there are K or less foreground tasks in the system, any task departure will force all other foreground tasks to depart from the system.
- When the system has some background tasks waiting and there are more than K foreground tasks, if a foreground task departure would create an idle server (e.g: there is no more foreground tasks for that particular server while there are still some foreground tasks in other servers), the task remains in the server for another phase of service and we force a task departure from one of the longest foreground task queues instead.

Informally, since a job's response time is a function of the last task's departure, the constraint on threshold d can force a job to complete earlier and thereby obtain a lower bound response time. The second and third departure constraints described above not only result in a lower bound on the job response time but combined with the trigger threshold $\overline{\mathbf{C}}$, this will create a tree structured transition diagram for M^l and we can use an approach similar to that developed in the last section to efficiently compute the job response time of M^l.

4.1 Proof for lower bound on expected job response time

In this section, we will prove that the model M^l provides a lower bound on the expected job response time. We will use the concept of *majorization* [21] in the proof so we first review some basic facts about majorization.

Definition 8 *$\boldsymbol{X}$ is said to majorize $\boldsymbol{Y}$ (written $\boldsymbol{X} \prec \boldsymbol{Y}$) iff*

$$\sum_{l=1}^{k} \hat{X}_l \;\leq\; \sum_{l=1}^{k} \hat{Y}_l, \quad k = 1, \ldots, K-1,$$

$$\sum_{l=1}^{K} \hat{X}_l = \sum_{l=1}^{K} \hat{Y}_l. \tag{21}$$

where $\hat{X}_l$ ($\hat{Y}_l$) is the $l-$largest component of $\boldsymbol{X}$ ($\boldsymbol{Y}$). If we replace the equality in (21) by

$$\sum_{l=1}^{K} \hat{X}_l \leq \sum_{l=1}^{K} \hat{Y}_l,$$

we obtain a weaker ordering. In this case we say that $\boldsymbol{Y}$ *weakly majorizes* $\boldsymbol{X}$ (written $\boldsymbol{X} \prec_w \boldsymbol{Y}$).

The following lemma states some properties regarding operations that can be performed on $\boldsymbol{X}$ and $\boldsymbol{Y}$ such that weak majorization is preserved.

Lemma 3 *Let $\boldsymbol{X}, \boldsymbol{Y} \in I\!N^K$ such that $\boldsymbol{X} \prec_w \boldsymbol{Y}$, then*

1. $(\hat{X}_1, \ldots, \hat{X}_k, \ldots, \hat{X}_l + 1, \ldots, \hat{X}_K) \prec_w (\hat{Y}_1, \ldots, \hat{Y}_k + 1, \ldots, \hat{Y}_l, \ldots, \hat{Y}_K)$, *for* $1 \leq k \leq l \leq K$
2. $(\hat{X}_1, \ldots, (\hat{X}_k - 1)^+, \ldots, \hat{X}_l, \ldots, \hat{X}_K) \prec_w (\hat{Y}_1, \ldots, \hat{Y}_k, \ldots, (\hat{Y}_l - 1)^+, \ldots, \hat{Y}_K)$, *for* $1 \leq k \leq l \leq K$

Proof. The proof follows in a straightforward manner from the definition of "$\prec_w$". The reader is referred to [21] for a detailed proof. ∎

In the following definition, we introduce the concept of a *Schur-convex* function, which will be useful in applying stochastic comparison based on majorization.

Definition 9 *A function $\phi : I\!N^K \rightarrow I\!R$ is said to be Schur-convex iff*

$$\phi(\boldsymbol{X}) \leq \phi(\boldsymbol{Y}), \quad \forall \boldsymbol{X}, \boldsymbol{Y} \in I\!N^K \qquad \textit{such that } \boldsymbol{X} \prec \boldsymbol{Y}.$$

Definition 10 *If $\boldsymbol{X}, \boldsymbol{Y} \in I\!N^K$ are random variables, then we say $\boldsymbol{X}$ is smaller than $\boldsymbol{Y}$ in the sense of Schur-convex order (written $\boldsymbol{X} \leq_{scx} \boldsymbol{Y}$) iff*

$$\phi(\boldsymbol{X}) \leq_{st} \phi(\boldsymbol{Y}), \qquad \forall \textit{Schur-convex}\, \phi.$$

If the class of functions is restricted to be increasing Schur-convex, then we say that $\boldsymbol{X}$ is smaller than $\boldsymbol{Y}$ in the sense of increasing Schur-convex order ($\boldsymbol{X} \leq_{iscx} \boldsymbol{Y}$).

From [21], we have a property of these orderings as expressed in the following lemma.

Lemma 4 *Let $\boldsymbol{X}, \boldsymbol{Y} \in I\!N^K$ be vector valued random variable's such that:*

$$\boldsymbol{X} \leq_{scx} \boldsymbol{Y} \quad (\boldsymbol{X} \leq_{iscx} \boldsymbol{Y})$$

There exist two random variable's $\widetilde{\boldsymbol{X}}$ and $\widetilde{\boldsymbol{Y}}$ such that:

$$\widetilde{\boldsymbol{X}} =_{st} \boldsymbol{X},\ \widetilde{\boldsymbol{Y}} =_{st} \boldsymbol{Y} \ \textit{and} \ \widetilde{\boldsymbol{X}} \prec \widetilde{\boldsymbol{Y}} \ (\widetilde{\boldsymbol{X}} \prec_w \widetilde{\boldsymbol{Y}})$$

almost surely.

With this lemma, we can couple the behavior of models M and M^l so that we can concentrate on comparing deterministic vectors. Letting $\boldsymbol{N}(t)$ ($\boldsymbol{N}^l(t)$) be the joint queue length for the model M (M^l) at time t. We have the following theorem:

Theorem 5 *If $\boldsymbol{N}^l(0) \leq_{iscx} \boldsymbol{N}(0)$, then $\boldsymbol{N}^l(t) \leq_{iscx} \boldsymbol{N}(t)$ for $\forall\, t \geq 0$*

Proof: Using Lemma 4, we first couple the initial queue length of these two models such that $\boldsymbol{N}^l(0) \prec_w \boldsymbol{N}(0)$. Now condition on the arrival times so that we can examine the two systems with exactly the same arrival sample path. For each server, we associate a *service event process* which is a Poisson process with parameter $k\mu$, where k is the number of stages for the task service Erlang distribution. Whenever a service event occurs associated with a server, a departure occurs if there is one or more customer in the queue at the time of the event. Note that the coupling of the service event times at the different servers is only possible if the service times at the servers are all mutually independent sequences of i.i.d exponential random variables with the same parameter[8].

Let $\{t_i\}_{n=0}^{\infty}$ be the sequence of times at which arrivals or service events occur ($t_0 \equiv 0$). We will establish the relation $\boldsymbol{N}^l(t) \prec_w \boldsymbol{N}(t)$ by induction on the event times.

Basis step. This follows from the coupling of the initial queue lengths.

Inductive step. Assume that $\boldsymbol{N}^l(t) \prec_w \boldsymbol{N}(t)$ for $t < t_i$. We will establish that $\boldsymbol{N}^l(t_i) \prec_w \boldsymbol{N}(t_i)$. Now there are several cases:

Arrival Event. If the event is a job arrival, since it adds k sub-tasks to each server, the $\boldsymbol{N}^l(t_i) \prec_w \boldsymbol{N}(t_i)$ relationship is preserved (this is easily seen as a generalization of Lemma 3).

Service Event. There are several cases:

[8]This is the origin of the restriction of service time to Erlangian service variables.

- *Normal task departure:* If a service event occurs at the i^{th} server and both systems, M and M^l, allow this departure (i.e., the i^{th} server is busy in both systems), $\prec_w$ is clearly preserved by Lemma 3, Property 1.
- *Constraint on threshold d:* If the lower bound model M^l disallows the task departure from the i^{th} server and forces a task departure from the longest queue due to threshold d constraint, then Property 2 of Lemma 3 can be applied to see that the $\prec_w$ relationship is preserved.
- *Constraint on task departures when there are background tasks:* In the lower bound model M^l, we have the constraint that if there are background tasks, we disallow any task departure which will create an idle server, by replacing the original departure with either a departure of a foreground task from the longest queue or a flush of all foreground tasks. In either cases, we see that by Property 2 of Lemma 3, the $\prec_w$ relationship is preserved.

This completes the induction step and thus we have $\boldsymbol{N}^l(t) \prec_w \boldsymbol{N}(t)$, $t \geq 0$. By the definition of weak majorization ($\prec_w$), this implies that $f(N^l(t)) \leq f(N(t))$ for any increasing Shur-convex function $f(t)$. Removing the conditioning on the arrival times and service times, we have

$$\boldsymbol{N}^l(t) \leq_{iscx} \boldsymbol{N}(t) \quad \forall t > 0$$

■

Corollary 3 *if $\boldsymbol{N}^l(0) \leq_{iscx} \boldsymbol{N}(0)$, then $N^l(t) \leq_{st} N(t)$, for $t \geq 0$.*

Proof: This follows from the preceding theorem and the fact that the function f defined in Equation (5) is a Shur-convex function. ■

4.2 Computational procedure for lower bound model M^l

In this section, we discuss the computational procedure for obtaining the expected response time for the lower bound model M^l. We first describe the class of arrival process we allow in our computational algorithm for the upper bound model M^l.

Let τ denote the random variable equal to the interarrival time of jobs to the system and let $A^*(s)$ denote the Laplace transform of its distribution. We consider a series-parallel stages distribution where each stage represents an exponentially distributed random variable τ_{ij} with parameter λ_{ij}. Without loss of generality, the class of interarrival distributions has the following form:

$$\begin{aligned} A^*(s) &= \sum_{j=1}^{R} \alpha_j \prod_{i=1}^{r_j} \frac{\lambda_{ij}}{s+\lambda_{ij}} && \text{for } 0<\alpha_j<1, \sum_{j=1}^{R}\alpha_j = 1 \text{ , } \lambda_{ij}>0 && (22)\\ \text{and} \qquad r_i &\le r_1 && \text{for } 2 \le i \le R && (23)\\ \lambda_{i1} &\le \lambda_{ij} && \text{for } 2 \le j \le R \text{ and } 1 \le i \le r_j && (24) \end{aligned}$$

such that $-\frac{dA^*(s)}{ds}|_{s=0} = \lambda^{-1}$. Note that because of Equations (23) and (24), the class of distributions allowed is more restrictive than the interarrival distribution of the upper bound model M^u. But even with these constraints, this class of interarrival distributions can accommodate a large class of distributions such as exponential, Erlang, hyperexponential, ..., etc. Figure 5 illustrates this type of series-parallel distribution.

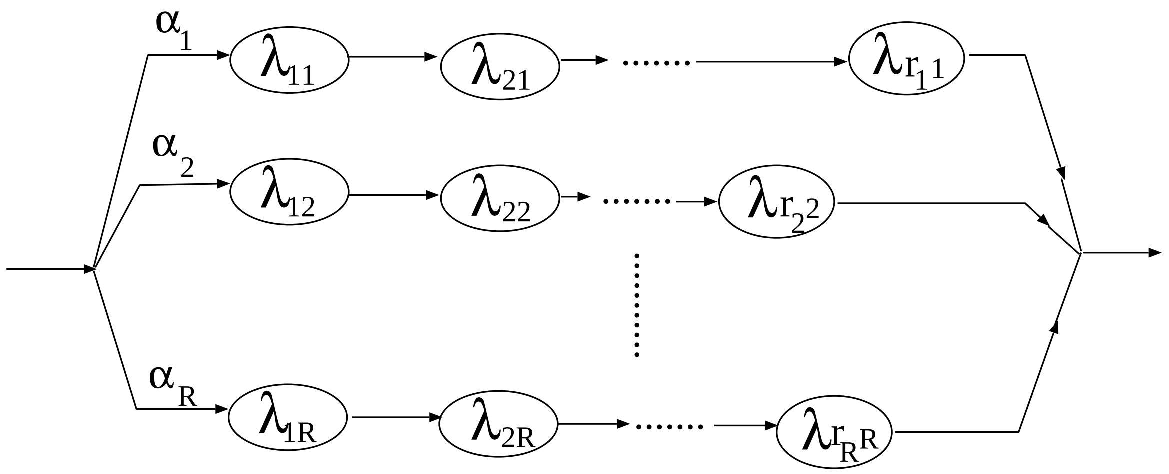

Figure 5: Series-parallel distribution for M^l.

To describe the computational procedure for the lower bound model M^l, it is important that we first describe the state space of the lower bound model. First, let us define the following:

Definition 11 *Let $S^l_{i,j}$, $i \ge 0, 0 \le j \le B^i - 1$, be the set of states in the lower bound model M^l where $S^l_{i,j}$ contains all states that are (1) in the i^{th} (where $i>0$) nested level of busy cycles, (2) the difference in the waiting queue lengths of foreground tasks among different servers is less than or equal to d, and (3) the i^{th} nested busy cycle was entered from the $(i-1)^{th}$ nested busy cycle through the trigger state $\mathcal{S}_{tt}(j \bmod B) \in S^l_{i-1,j/B}$.*

Transitions between states in $S^l_{i,j}$ behaves like the original model M except :

- For task departures that would violate the threshold d, the task has to remain in the server for another phase of service and the system forces a foreground task departure from one of the longest queue.
- If a foreground task departure would create an idle server (e.g., there are no more foreground tasks for that particular server), the task has to remain in the server for another phase of service and the system forces a task departure from one of the longest foreground task queues.
- If the system has some background tasks waiting and there are K or less foreground tasks in the system, any task departure will force all foreground tasks to depart from the server.
- If a new job arrives and the system is in one of the trigger states, for example $\mathcal{S}_{tt}(l)$, then a new busy cycle begins with this newly arrived job and the system enters state $(1, [0, k], \cdots, [0, k])$ in $S^l_{i+1,jB+l}$.
- Upon completion of the last foreground job in $S^l_{i,j}$, the previously suspended set of tasks is released for service. That is, the system will make a transition back to one of the trigger states in $S^l_{i-1,j/B}$.

With this definition, we can see again that the state space of the lower bound model can be organized as a tree with each node of the tree containing the states in $S^l_{i,j}$. Figure 6 illustrates the lower bound model M^l for Poisson arrival, exponential service time and $K = 2, d = 2, C = 5$. Also note that although the definition of $S^l_{i,j}$ and $S^u_{i,j}$ are similar, the *internal transition structures are different.*

As stated above, upon departure of the last foreground job in $S^l_{i,j}$, there will be a transition back to one of the trigger states in $S^l_{i-1,j/B}$. At this point, similar to what was done in the upper bound model M^u, we change the arrival process so that we can have an efficient algorithm for solving the lower bound model. First, we define random variables $\hat{\tau}_{kj}$ as:

$$\hat{\tau}_{kj} = \sum_{l=k}^{r_j} \tau_{lj} \qquad \text{for } 1 \le j \le R \text{ and } 1 \le k \le r_j \tag{25}$$

and the Laplace transform of $\hat{\tau}_{kj}$ as:

$$\hat{A}^*_{kj}(s) = \prod_{l=k}^{r_j} \left(\frac{\lambda_{lj}}{s + \lambda_{lj}} \right)$$

We can also express $A^*(s)$ as:

$$A^*(s) = \sum_{j=1}^{R} \alpha_j \hat{A}^*_{1j}(s)$$

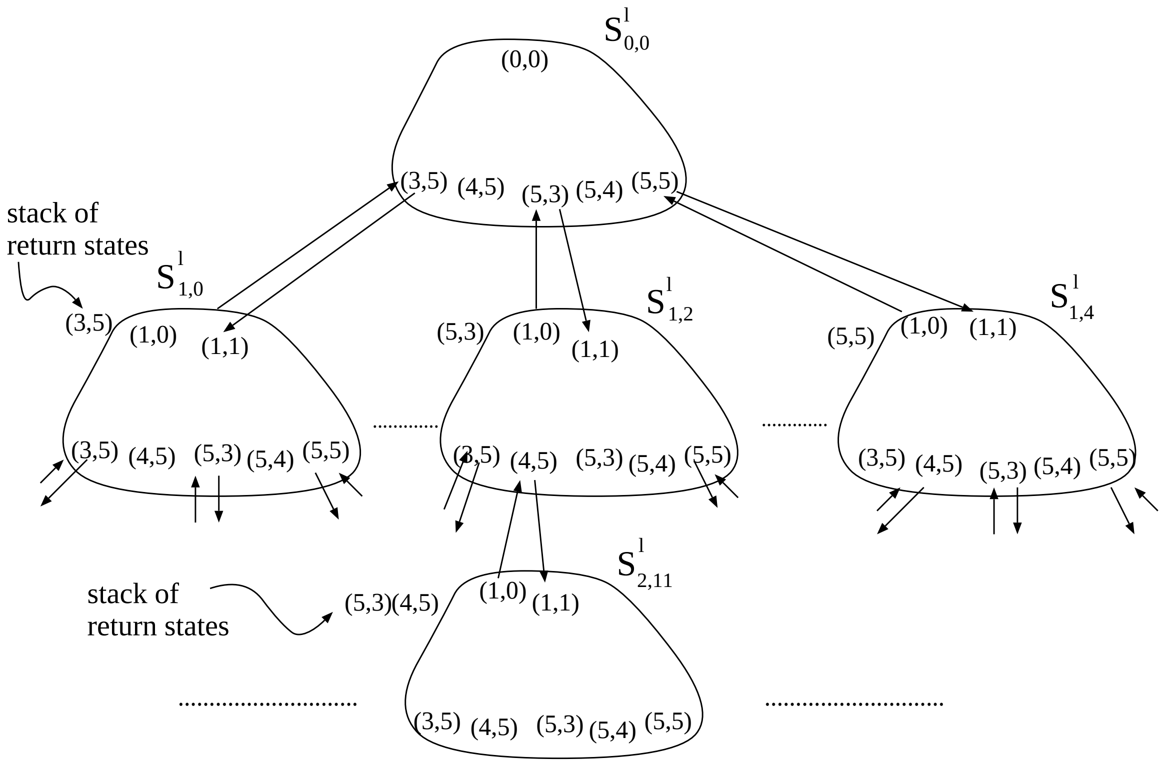

Figure 6: State Space Partition for Lower Bound Model M^l.

We have the following observation:

Theorem 6 $\tau \leq_{st} \hat{\tau}_{11}$.

Proof: Based on Equation (24), we have

$$\lambda_{ij} \geq \lambda_{i1} \qquad \text{for } 2 \leq j \leq R \text{ and } 1 \leq i \leq r_j$$

therefore

$$\lambda_{ij} \leq_{st} \lambda_{i1} \qquad \text{for } 2 \leq j \leq R \text{ and } 1 \leq i \leq r_j$$

We then have the following observations:

$$\sum_{k=i}^{r_j} \tau_{kj} \quad \leq_{st} \quad \sum_{k=i}^{r_j} \tau_{k1} \qquad \text{for } 2 \leq j \leq R \text{ and } 1 \leq i \leq r_j$$

$$\hat{\tau}_{1j} \quad \leq_{st} \quad \sum_{k=1}^{r_j} \tau_{k1} \quad \leq_{st} \quad \hat{\tau}_{11} \qquad \text{for } 2 \leq j \leq R$$

$$\tau = \sum_{j=1}^{R} \alpha_j \hat{\tau}_{1j} \quad \leq_{st} \quad \sum_{j=1}^{R} \alpha_j \hat{\tau}_{11} = \hat{\tau}_{11}$$

∎

Let Z be a random variable that takes on value of 0 or 1. Define

$$\hat{\tau}^* = \begin{cases} \tau & \text{Prob}(Z=1) \\ \hat{\tau}_{11} & \text{Prob}(Z=0) \end{cases}$$

Corollary 4 $\tau \leq_{st} \hat{\tau}^*$.

Proof: Since

$$\text{Prob}(Z=1)\tau + \text{Prob}(Z=0)\tau \leq_{st} \text{Prob}(Z=1)\tau + \text{Prob}(Z=0)\hat{\tau}_{11}$$

therefore,

$$\tau \leq_{st} \hat{\tau}^*$$

■

Let $n_i = \{n_i(t), t \geq 0\}, i = 1, 2$, denote the process defined by the number of tasks at a server when the interarrival distribution is G_τ and $F_{\hat{\tau}^*}$ respectively. We want to show that

$$\{n_2(t), t \geq 0\} \leq_{st} \{n_1(t), t \geq 0\}.$$

Corollary 5 $\{n_2(t), t \geq 0\} \leq_{st} \{n_1(t), t \geq 0\}$.

Proof: The proof is similar to Theorem 3 and therefore is omitted in the interest of space. ■

The significance of Theorems 6, Corollary 4 and Corollary 5 is that at some *imbedded time*, we can modify the interarrival distribution according to Theorem 6 and we can have a lower bound on the number of tasks in the system. These imbedded times are defined to be the times when all the foreground tasks are completed and the system is switching to the last suspended set of tasks (which then becomes the foreground tasks). When this occurs, the next job arrival occurs with an interarrival time $\hat{\tau}_{11}$. After this arrival, the original interarrival process is again in effect.

The approach to compute the expected job response time T^l for the lower bound model M^l is similar to the approach we took in obtaining the T^u in M^u, that is:

1. compute the conditional state probability given that the system is in set $S^l_{i,j}$.

2. aggregate each $S^l_{i,j}$ into a single state $s^l_{i,j}$
3. compute the transition rates between aggregate states.
4. given the conditional state probabilities of $S^l_{i,j}$ and aggregate state probabilities $s^l_{i,j}$, applying Equation (4) and Equation (5), we can find the T^l, the expected job response time for lower bound model M^l.

First, we show how to obtain the conditional state probability given that the system is in set $S^l_{i,j}$.

Theorem 7 *Consider a Markov process with rate matrix $\hat{Q}_{S^l_{i,j},S^l_{i,j}}$, which is equal to $Q_{S^l_{i,j},S^l_{i,j}}$ except for the following modification:*

1. *A transition to the parent set of states, $S^l_{i-1,k}$, is changed to a transition to the initial state, which is the state where $S^l_{i-1,k}$ is entered.*
2. *Each transition to a child in set $S^l_{i+1,l}$ is changed to the return state from the child in set $S^l_{i+1,l}$.*

The steady state probability vector for the Markov process with rate matrix $\hat{Q}_{S^l_{i,j},S^l_{i,j}}$ is the conditional state probability vector given the system is in set $S^l_{i,j}$.

Proof: The proof is similar to the proof in Theorem 4. ∎

With the above theorem, we can aggregate each set of states $S^l_{i,j}$ as a single state $s^l_{i,j}$. After aggregation, the state space is depicted in Figure 7. The transitions between aggregates states can be computed as follow:

$$\begin{aligned}
\lambda^{l'}_i &= \hat{\pi}'(a)\lambda && for\, a \in \mathcal{S}_{tt}\, where\, \mathcal{S}_{tt} \subset S^l_{0,0}, 0 \le i \le B-1 \quad (26)\\
\lambda^{l*}_i &= \hat{\pi}(a)\lambda && for\, a \in \mathcal{S}_{tt}\, where\, \mathcal{S}_{tt} \subset S^l_{1,0}, 0 \le i \le B-1 \quad (27)\\
\nu_l &= \sum_{s\in\Psi} \hat{\pi}(s)\mu && (28)
\end{aligned}$$

where Ψ be the set of states in $S^l_{i,j}$, $i \ge 1$, such that there are transitions from Ψ back to parents of $S^l_{i,j}$. $\hat{\pi}'(a)$ and $\hat{\pi}(a)$ are the steady state probability for Markov processes with rate matrix $\hat{Q}_{S^l_{0,0},S^l_{0,0}}$ and $\hat{Q}_{S^l_{1,0},S^l_{1,0}}$ respectively.

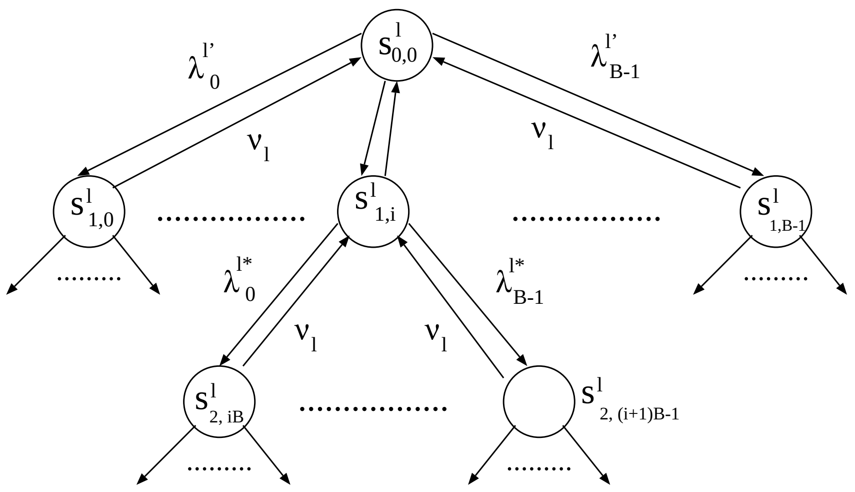

Figure 7: Aggregated Process for lower bound model M^l.

Let $\pi^*(0,0)$ be the steady state solution for the aggregated state $s^l_{0,0}$. It is equal to:

$$\pi^*(0,0) \quad = \quad \frac{\nu_l - \lambda^{l*}}{\nu_l + \lambda^{l'} - \lambda^{l*}} \tag{29}$$

To compute for the expected job response time T^l for the lower bound model M^l, let $r^l_{f_0}$ be the expected number of tasks given the system is in $S^l_{0,0}$, r^l_f be the expected number of foreground tasks given the system is in $S^l_{i,j}$ for $i \geq 1$ and, $r^l_b(i)$ be the expected number of background tasks given the system is in $S^l_{i,j}$ for $i \geq 1$. Therefore, given the rate matrix $\hat{Q}_{S^l_{0,0},S^l_{0,0}}$ and $\hat{Q}_{S^l_{i,j},S^l_{i,j}}$, we can compute $r^l_{f_0}$ and r^l_f. The number of background tasks given the system is in i^{th} level of busy cycle is equal to iKC. Therefore, the total number of tasks in the lower bound model M^l is:

$$N^l \quad = \quad \left(\frac{\nu_l - \lambda^{l*}}{\nu_l + \lambda^{l'} - \lambda^{l*}}\right)\left[r^l_{f_0} + \frac{r^l_f\,\lambda^{l'}}{\nu_l - \lambda^{l*}} + \frac{\lambda^{l'}KC\nu_l}{(\nu_l - \lambda^{l*})^2}\right] \tag{30}$$

and the expected job response time T^l for the lower bound model M^l is:

$$T^l \quad = \quad \left[\frac{\nu_l - \lambda^{l*}}{K\lambda\,(\nu_l + \lambda^{l'} - \lambda^{l*})}\right]\left[r^l_{f_0} + \frac{r^l_f\,\lambda^{l'}}{\nu_l - \lambda^{l*}} + \frac{\lambda^{l'}KC\nu_l}{(\nu_l - \lambda^{l*})^2}\right] \tag{31}$$

5 Applications and Numerical Examples

In this section, we present two examples in order to illustrate the applicability of the bounding algorithm.

Let us consider a computer vision system implemented in a homogeneous workstation cluster under the *Parallel Virtual Machines (PVM)* environment [27]. An image will arrive to the computer vision control manager, which will then assign a portion of the image to each workstation for computation. Each workstation will in turn perform some ray-tracing calculation for its part of the image and the total image is ready for display when all workstations have finished their ray-tracing operations. This type of application can be mapped to the fork-join paradigm and the environment is illustrated in Figure 8.

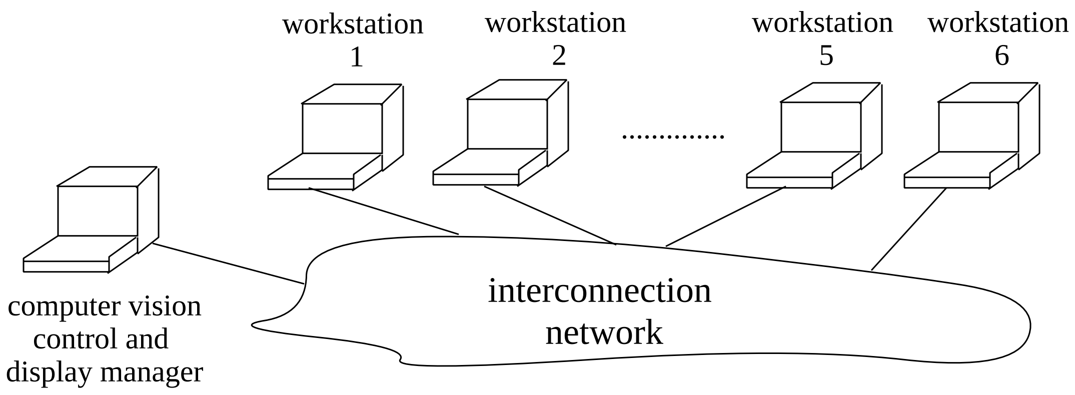

Figure 8: Computer Vision Program in Workstation Cluster Environment.

Assume that there are 6 workstations ready for ray-tracing operations The job arrival distribution is Poisson and the service time distribution of each task is exponential with rate equal to 1.0. We vary the utilization of the system from 0.1 to 0.9 by changing the arrival rate. Table 1 gives the upper and lower bound of the expected job response time. Percentage error is defined to be $\frac{R_u - R_l}{R_u + R_l}$ x 100%.

The second scenario we illustrate is the same system with 6 servers but the job arrival process is a Erlang−2 distribution. The service time distribution of tasks are exponential with rate 1.0. Again, we vary the input rate so the utilization of the system can varies from 0.1 to 0.9. Table 2 illustrates the upper and lower bound for the expected job response time.

The third system we illustrate is similar to the second system except the arrival process is Poisson with rate λ but the service time distribution is Erlang−2 distribution. Table 3 illustrates the upper and lower bound for the expected job response time.

For the last example, we want to illustrate the tradeoff between computational cost and accuracy of the bounds. Let us consider the system where the job arrival process is Poisson and the task service time is Erlang−2 distribution. By fixing the system utilization at 0.7 and increasing the number of states generated, we see the improvement of the bounds on the mean response time. The results are illustrated in Table 4

System Utilization	T^u	T^l	Spread of Bounds	C	d	States Generated	Percentage Error
0.1	2.64	2.67	0.03	8	5	1218	0.56 %
0.2	2.91	2.96	0.05	8	5	1218	0.85 %
0.3	3.26	3.32	0.06	8	5	1218	0.91 %
0.4	3.73	3.81	0.08	8	6	1848	1.06 %
0.5	4.35	4.45	0.10	8	6	1848	1.14 %
0.6	5.31	5.44	0.13	9	7	3300	1.21 %
0.7	7.00	7.20	0.20	9	7	3300	1.41 %
0.8	10.21	10.52	0.31	10	7	4092	1.49 %
0.9	19.80	20.33	0.63	11	8	6864	1.57 %

Table 1: Poisson Arrival and Exponential Service Time.

System Utilization	T^u	T^l	Spread of Bounds	C	d	States Generated	Percentage Error
0.1	2.50	2.52	0.02	6	4	924	0.40 %
0.2	2.65	2.68	0.03	6	4	924	0.56 %
0.3	2.92	2.97	0.05	6	4	924	0.85 %
0.4	3.21	3.30	0.09	6	5	1428	1.38 %
0.5	3.74	3.85	0.11	6	5	1428	1.45 %
0.6	4.58	4.72	0.14	6	5	1428	1.50 %
0.7	5.70	5.89	0.19	7	5	1932	1.64 %
0.8	8.40	8.70	0.30	7	5	1932	1.75 %
0.9	16.22	16.87	0.65	8	6	3696	1.96 %

Table 2: 2-stage Erlang Arrival and Exponential Service Time.

6 Conclusion

Fork-Join queueing is one of the basic parallel computational models building blocks. Due to correlation among servers and infinite queueing capacities for each server, no closed-form solution exists for the general case. The problem cannot be solved by direct numerical computation due to its infinite state space. In this paper, we propose an algorithm to obtain the upper and lower bound of the expected response time of a job. The algorithm also provides the flexibility to tradeoff computational resources and tighter

System Utilization	T^u	T^l	Spread of Bounds	C	d	States Generated	Percentage Error
0.1	2.14	2.16	0.02	6	4	5440	0.46 %
0.2	2.31	2.34	0.03	6	4	5440	0.65 %
0.3	2.53	2.57	0.04	6	4	5440	0.78 %
0.4	2.82	2.87	0.05	6	4	5440	0.88 %
0.5	3.29	3.37	0.08	6	5	8568	1.20 %
0.6	3.90	4.00	0.10	6	5	8568	1.52 %
0.7	4.90	5.07	0.17	7	5	11592	1.72 %
0.8	7.01	7.27	0.26	7	6	16632	1.82 %
0.9	14.70	15.29	0.59	8	7	30096	1.97 %

Table 3: Poisson Arrival and 2-stage Erlang Service Time.

C	d	States Generated	Response Time Upper Bound(T^u)	Response Time Lower Bound(T^l)	Spread of Bounds	Percentage Errors
7	5	11592	4.80	5.25	0.44	4.38 %
7	6	16632	4.90	5.07	0.17	1.72 %
8	7	30096	4.97	5.05	0.08	0.80 %

Table 4: Computational Cost vs. Accuracy.

bounds. There is ongoing work to investigate the possibility of a priori determining d and C_i to obtain specified error bounds.

References

[1] F. Baccelli and A.M. Makowski. *Simple computable bounds for the fork-join queue.* Proceeding 19^{th} Annual Conference of Information Sciences and Systems. The John Hopkins University, Baltimore, MD, pp. 436-441,March 1985.

[2] F. Baccelli, A.M. Makowski and A. Shwartz. *Simple computable bounds and approximations for fork-join queue.* Intern. Workshop on Computer Performance Evaluation, Tokyo, September 1985, pp.437-450.

[3] F. Baccelli, A.M. Makowski and A. Shwartz. *The Fork-Join Queue and Related Systems with Synchronization constraints: Stochastic Ordering and Computable Bounds.* Adv. Applied Probability, Vol 21, pp. 629-660, 1989.

[4] F. Baccelli, W.A. Massey and D. Towsley. *Acyclic Fork-Join Queueing Networks.* Journal of ACM, Vol.36, No. 3, pp. 615-642, July 1989.

[5] A. Beguelin, J.J. Dongarra, A. Geist, R. Manchek, V.Sunderam. *PVM and HeNCE: Tools for Heterogeneous Network Computing.* Advances in Parallel Computing: Environments and Tools for Parallel Scientific Computing, Elsevier Science Publishers, pp. 139-153, 1993.

[6] S. Chen and D. Towsley. *Design and Modeling Policies for Two Server Fork/Join Queueing Systems.* University of Mass. COINS Technical Report 91-39, 1991.

[7] P. J. Courtois. *Decomposability — queueing and computer system applications.* Academic Press, New York, 1977.

[8] P. J. Courtois, P. Semal. *Computable Bounds for Conditional Steady-State Probabilities in Large Markov Chains and Queueing Models.* IEEE JSAC, Vol 4, number 6, September, 1986.

[9] L. Fatto and S. Hahn. *Two parallel queues created by arrivals with two demands I*, SIAM J. Appl. Math., Vol. 44, pp.1041-1053, Oct, 1984.

[10] L. Fatto. *Two parallel queues created by arrivals with two demands II*, SIAM J. Appl. Math., Vol. 45, pp.861-878, Oct, 1985.

[11] G.A. Geist, V.S. Sunderam. *The PVM System: Supercomputing Level Concurrent Computations on Heterogeneous Network of Workstations.* The Sixth Distributed memory Computing Conference Proceedings, pp. 258-261, April 1991.

[12] P. Heidelberger and K.S. Trivedi. *Queueing Network models for parallel processing with asynchronous tasks.* IEEE Trans. Computer, Vol. C-31, pp.1099-1109, Nov, 1982.

[13] P. Heidelberger and K.S. Trivedi. *Analytic queueing models for programs with internal concurrency.* IEEE Trans. Computer, Vol. C-32, pp.73-82, Nov, 1983.

[14] C.A.R. Hoare. *Communicating Sequential Processes.* Prentice-Hall International, London, 1985.

[15] F.P. Kelly. *Reversibility and Stochastic Networks.* Wiley, New York, 1979.

[16] L. Kleinrock. *Queueing Systems Volume I: Theory.* Wiley International, 1975.

[17] P. Konstantopoulos and J. Walrand. *Stationary and Stability of Fork-Join Networks.* Journal of Applied Probability, Vol.26, pp.604-614, 1989.

[18] C.P. Kruskal and A. Weiss. *Allocating independent substasks on parallel processors.* IEEE Trans. Software Eng. SE-11, pp. 1001-1016, October. 1985

[19] John C.S. Lui, R.R. Muntz and D. Towsley. *Bounding the Response Time of a Minimum Expected Delay Routing System: An Algorithmic Approach* Submitted for publication in IEEE Trans. on Computers.

[20] John C.S. Lui, R.R. Muntz. *Computing Bounds on Steady State Availability of Repairable Computer Systems.* Journal of ACM, Vol 41, No. 4, pp. 676-707, July 1994.

[21] Marshall and Olkin. *Inequalities: Theory of Majorization and Applications*, Academic Press, New York, 1979.

[22] A. Makowski and S. Verma. *Interpolation Approximations for Symmetric Fork-Join Queues.* Performance '93.

[23] R.R. Muntz, E. De Souza E Silva and A. Goyal. *Bounding Availability of Repairable Computer Systems.* In Proceedings of 1989 ACM SIGMETRICS and PERFORMANCE '89, also in special issue of IEEE Trans. Computers, Dec. 1989, pp. 19-30.

[24] R. Nelson and A.N. Tantawi. *Approximate Analysis of Fork/Join Synchronization in Parallel Queues.* IEEE Transaction of Computers, Vol. 37, No. 6, June, 1988.

[25] R. Nelson, D. Towsley and A.N. Tantawi. *Performance Analysis of Parallel Processing Systems.* IEEE Transaction of Software Engineering, 14, pp.532-540, April 1988.

[26] I.C. Pyle. *The Ada Programming Language.* Prentice-Hall International, London, 1981.

[27] A. Giest, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, V. Sunderam. *PVM* 3.0 *User's Guide and Reference Manual.* Oak Ridge National Laboratory, 1993.

[28] S.M. Ross. *Introduction to Probability Models.* Academic Press, 1970.

[29] M. R. Stonebraker. *The Case for Shared-Nothing.* Proceeding of the 1986 Data Engineering Conference. IEEE, 1986.

[30] D. Towsley, J.A. Rommel and J.A. Stankovic. *The Performance of Processor Sharing Scheduling Fork-Join in Multiprocessors.* High-Performance Computer Systems. E. Gelenbe. ed. North-Holland, Amsterdam, pp. 146-156, 1988.