A DECOMPOSITION PROCEDURE FOR THE ANALYSIS OF A CLOSED FORK/JOIN QUEUEING SYSTEM

by

Y. C. Liu

Department of Electrical & Computer Engineering

Center for Communication and Signal Processing

and

H. G. Perros

Department of Computer Science

Center for Communication and Signal Processing

North Carolina State University

Raleigh, NC 27695-8206

CCSP-TR-88/18

June 1988

Abstract

An iterative approximation algorithm for analyzing a closed queueing system with a K-sibling fork/join queue is presented. The iterative procedure is based on a combination of nearly complete decomposability and the Gauss-Seidel method. The approximation procedure gives good results for the mean response time and the system throughput. However, it gives results which are an upper bound of the mean response time of the fork/join operation and a lower bound of the system throughput, for both homogeneous and non-homogeneous cases. Furthermore, the iterative procedure converges to the exact solution in the case of the closed 3-sibling fork/join queue.

1. Introduction

There is currently considerable interest in the development of tools for analyzing the performance of distributed and parallel processing systems. In such systems, quite often, a job is split into two or more sub-jobs. These sub-jobs execute independently of one another, and at the end of their execution, they recompose to the original job. These type of operation is known as fork/join, or disassembly/assembly in production system. They occur in multiprocessor computing systems, distributed database systems, telecommunication systems and flexible manufacturing systems.

In this paper, we consider a closed queueing system comprising of a fork node and a K-sibling fork/join queue ($K \geq 2$) with M jobs. When a job finishes its service at the fork node, it is split into a fixed number K of sub-jobs called siblings. Each sibling joins a different sibling queue, where it executes independently of the other siblings. Upon completion of its service, a sibling enters the synchronization queue where it waits for the other siblings. As soon as all the siblings have been served, they merge into the original job which immediately joins the fork node. The time elapsing between the fork and the join operation of a job is called the **response time**. Also, the time a sibling spends waiting for the other siblings is referred to as the **synchronization delay**.

Queueing networks with fork/join operations are in general difficult to analyze. An exact numerical analysis of a fork/join queue results in an explosion of the state space which renders it computationally intractable. Also, in general,

fork/join models do not have analytically closed form solutions. In view of this, most of the fork/join systems reported in the literature have been analyzed using various approximation methods.

Flatto [9,10] considered an open fork/join system, assuming that jobs arrive in a poisson fashion. Upon arrival, a job is split into two siblings, each sibling is served by a different server. The service time at each server is exponentially distributed with a different mean. As soon as the two siblings have been served, they recompose into an original job which leaves the system immediately. Flatto obtained the stationary distribution of the response time, and for each queue he obtained the queue length distribution and its expectation conditioned upon the other queue. Nelson and Tantawi [16] proposed an approximation technique, called the scaling approximation. They assumed that the mean response time increases at the same rate as the number of siblings increases. They derived a closed-form approximate expression of the mean response time for an open fork/join queueing system consisting of K ($K \geq 2$) identical servers. Mailles [15] obtained bounds of the mean response time for the same system and also for various series-parallel fork/join networks. The case of general service time was considered by Baccelli and Makowski [1]. Their analysis was based on renewal type of arguments. They concluded that an upper bound of the response time can be obtained using a GI/G/1 mutually independent parallel queueing system and a lower bound can be obtained using a D/G/1 parallel queueing system. The tightness of these bound was not analyzed. In [2], Baccelli, Makowski and Schwartz used two bounding

methodologies to derive a simple lower and upper bound of the response time. The first approach was based on the convex increasing ordering. The second approach was based on the notion of associated random variables, a method similar to the one used by Nelson and Tantawi [16]. In [3], Baccelli, Massey and Towsley extended these ordering and bounding techniques to analyze acyclic fork/join queueing networks. They obtained bounds for the network response time.

Heidelberger and Trivedi [11] considered a closed queueing network model of a computing system in which jobs divide into two or more asynchronous tasks, i.e., synchronization between tasks is not required. They developed an iterative technique for solving a sequence of product-form type queueing networks. In [12], they extended the model to include a join node. Two approximation methods were developed. The first one was based on a decomposition approximation consisting of an inner model, a product-form queueing network, and an outer model, a finite state markov chain. The other approximation was based on the complementary delays method, which iteratively solves a sequence of product form queueing networks. The fork/join queue was analyzed numerically as a closed network by Duda and Czachorski [7]. The numerical approach is inherently limited to small problems. Also, Duda [8] developed an approximation algorithm for analyzing a series-parallel fork/join networks by constructing an approximately equivalent queueing network with a product-form solution. Liu and Perros [14] analyzed approximately a closed queueing system with a K-sibling fork/join queue using decomposition and aggregation. They iteratively reduced the K-sibling fork/join

queue into a 2-sibling fork/join queue, which was then used to replace the K-sibling fork/join queue in the original queueing network. This queueing system was then analyzed numerically. They also developed a modification procedure for the homogeneous case in order to improve the accuracy of the approximation algorithm.

In this paper, we consider the closed queueing system comprising of a fork node and a K-sibling fork/join queue ($K \geq 2$), as shown in figure 1. The fork/join queue may consist of homogeneous or non-homogeneous exponential servers, i.e., they may all have the same service rate or different service rates. Let μ_i be the service rate at sibling queue i, i = 1, 2,...,K, and μ_0 be the service rate at the fork node. The buffer size of each queue is assumed to be infinite. We analyze this model by first studying the closed K-sibling fork/join queue shown in figure 2. This model is obtained from the original system by shorting out the fork node. In the following section, we describe an approach for analyzing exactly this closed K-sibling fork/join queue with K = 3, i.e., with 3 siblings. This approach is based on a combination of nearly complete decomposability (see Courtois [5]) and the Gauss-Seidel method. In section 3, we use this approach to approximately reduce the closed K-sibling fork/join queue to a 2-sibling fork/join queue. Thus, the original system, shown in figure 1, can be approximately reduced to a model consisting of the fork node and a 2-sibling fork/join queue. This final model is analyzed using the same approach. In section 4, we compare the approximate results against exact numerical and simulation data. The approximation method gives

good results for the mean response time and the system throughput. However, the procedure gives a lower bound of the system throughput and an upper bound of the mean response time. Finally, the conclusions are given in section 5.

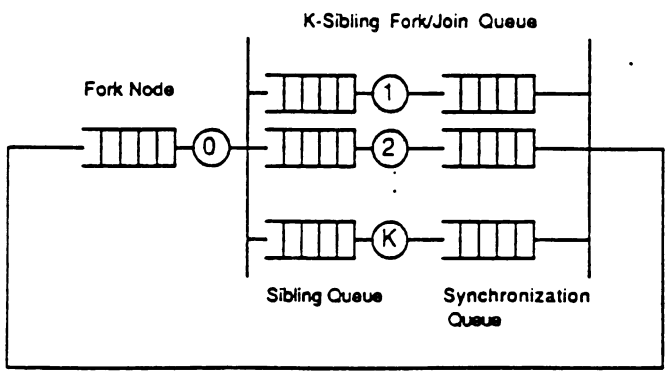

figure 1 : The system under study

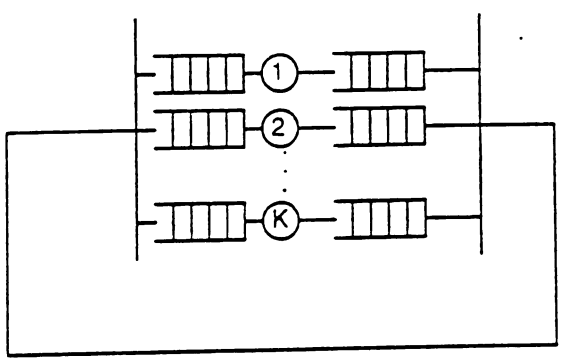

figure 2 : The K-sibling fork/join queue as a closed queueing network

2. Exact Solution of the Closed 3-Sibling Fork/Join Queue

In this section, we analyze the closed K-sibling fork/join queue with M jobs, shown in figure 2, assuming that K = 3. The state of the system is described by the vector (s_1,s_2,s_3), where s_i is the number of jobs at sibling queue i. We have $M = \max(s_1,s_2,s_3)$. The total number of states of the closed 3-sibling fork/join queue is equal to $(M+1)^3 - M^3$.

For each state (s_1,s_2,s_3), let $s' = \max(s_1,s_2)$. Now, let us partition the state space into groups, so that all the states in the same group have the same value of s'. For example, for M = 2 we have the following grouping of states.

(s_1,s_2,s_3)	(s',s_3)	index
(0,0,2)	(0,2)	0
(0,1,2) (1,1,2) (1,0,2)	(1,2)	1
(0,2,2) (1,2,2) (2,2,2) (2,1,2) (2,0,2)	(2,2)	2
(0,2,1) (1,2,1) (2,2,1) (2,1,1) (2,0,1)	(2,1)	1′
(0,2,0) (1,2,0) (2,2,0) (2,1,0) (2,0,0)	(2,0)	0′

table 1 : State space of the closed 3-sibling fork/join queue

The macro states ($s'.s_3$), in general, take the values : (i,M), i = 0, 1, 2,.....M, and (M, j), j = M-1, M-2.....0. These states are indexed as follows: 0, 1,....,M, $(M-1)'$, $(M-2)'$ ....,0'. The indexing of these states for the example of M =2 is given in table 1.

If the states are enumerated as shown in table 1, then the rate matrix $\mathbf{Q}(M)$ of the 3-sibling fork/join queue has a block tri-diagonal structure. For example, the rate matrix $\mathbf{Q}(M)$, for M = 2 is as follows:

$$\mathbf{Q}(M) = \begin{bmatrix} \mathbf{Q}_{00}(M) & \mathbf{Q}_{01}(M) & & & & \\ \mathbf{Q}_{10}(M) & \mathbf{Q}_{11}(M) & \mathbf{Q}_{12}(M) & & & \\ & \mathbf{Q}_{21}(M) & \mathbf{Q}_{22}(M) & \mathbf{Q}_{21'}(M) & & \\ & & \mathbf{Q}_{1'2}(M) & \mathbf{Q}_{1'1'}(M) & \mathbf{Q}_{1'0'}(M) & \\ & & & \mathbf{Q}_{0'1'}(M) & \mathbf{Q}_{0'0'}(M) \end{bmatrix}, \tag{1}$$

where,

$$\mathbf{Q}_{10}(M) = \begin{bmatrix} \mu_2 \\ \\ \mu_1 \end{bmatrix}, \tag{2}$$

$$\mathbf{Q}_{11}(M) = \begin{bmatrix} -a_1 & & \\ \mu_1 & -a_2 & \mu_2 \\ & & -a_3 \end{bmatrix}, \tag{3}$$

$$\mathbf{Q}_{12}(M) = \begin{bmatrix} 0 & \mu_3 & \\ & \mu_3 & \\ & & \mu_3 \; 0 \end{bmatrix}, \tag{4}$$

$$\mathbf{Q}_{21}(M) = \begin{bmatrix} \mu_2 & & & \\ & \mu_2 & & \\ & & \mu_1 & \\ & & & \mu_1 \end{bmatrix}, \tag{5}$$

$$\mathbf{Q}_{22}(M) = \begin{bmatrix} -a_1 & & & & & \\ \mu_1 & -a_2 & & & & \\ & \mu_1 & -a_2 & \mu_2 & & \\ & & & -a_2 & \mu_2 & \\ & & & & -a_3 & \end{bmatrix}, \tag{6}$$

$$\mathbf{Q}_{21'}(M) = \begin{bmatrix} \mu_3 & & & & \\ & \mu_3 & & & \\ & & \mu_3 & & \\ & & & \mu_3 & \\ & & & & \mu_3 \end{bmatrix}, \tag{7}$$

$a_1 = \mu_2 + \mu_3$, $a_2 = \mu_1 + \mu_2 + \mu_3$, $a_3 = \mu_1 + \mu_3$, $\mathbf{Q}_{1'2}(M) = \mathbf{Q}_{0'1'}(M) = (0,\mathbf{Q}_{21}(M),0)$, $\mathbf{Q}_{1'1'}(M) = \mathbf{Q}_{0'0'}(M) = \mathbf{Q}_{22}(M)$, and $\mathbf{Q}_{21'}(M) = \mathbf{Q}_{1'0'}(M)$, $\mathbf{Q}_{00}(M) = (-\mu_3)$, and $\mathbf{Q}_{01}(M) = (0,\mu_3, 0)$.

The procedure for obtaining the stationary vector involves the use of Courtois' procedure for nearly complete decomposable matrices [5] and the Gauss-Seidel method. In particular, we iterate between these two procedures starting with Courtois' method. We first construct a block-diagonal matrix $\mathbf{Q}^*(M)$ by adding the elements of the upper diagonal block into the corresponding elements of the diagonal block. The elements of the lower diagonal block are distributed by adding its first non-zero column to the second column of the diagonal block, its second non-zero column to the third column of the diagonal block, and so on. Thus, for

the example of M = 2, we have

$$\mathbf{Q}^*(M) = \begin{bmatrix} \mathbf{Q}^*_{00}(M) & & & & & \\ & \mathbf{Q}^*_{11}(M) & & & & \\ & & \mathbf{Q}^*_{22}(M) & & & \\ & & & \mathbf{Q}^*_{1'1'}(M) & & \\ & & & & \mathbf{Q}^*_{0'0'}(M) & \end{bmatrix}, \tag{8}$$

where

$$\mathbf{Q}^*_{11}(M) = \begin{bmatrix} -\mu_2 & \mu_2 & \\ \mu_1 & -(\mu_1+\mu_2) & \mu_2 \\ & \mu_1 & -\mu_1 \end{bmatrix}, \tag{9}$$

$$\mathbf{Q}^*_{22}(M) = \begin{bmatrix} -\mu_2 & \mu_2 & & & \\ \mu_1 & -(\mu_1+\mu_2) & \mu_2 & & \\ & \mu_1 & -(\mu_1+\mu_2) & \mu_2 & \\ & & \mu_1 & -(\mu_1+\mu_2) & \mu_2 \\ & & & \mu_1 & -\mu_1 \end{bmatrix}, \tag{10}$$

We note that the sub-matrix $\mathbf{Q}^*_{22}(M)$, is the rate matrix of the 2-sibling fork/join queue obtained by shorting sibling queue 3, with 2 jobs in the system. The sub-matrix $\mathbf{Q}^*_{11}(M)$ is the rate matrix of the same shorted 2-sibling fork/join queue, with 1 job in the system. The sub-matrix $\mathbf{Q}^*_{00}(M)$ is a singleton. Also, for the remaining sub-matrices we have $\mathbf{Q}^*_{1'1'}(M) = \mathbf{Q}^*_{0'0'}(M) = \mathbf{Q}^*_{22}(M)$. It is a known fact that the 2-sibling fork/join queue has a product form solution (see Duda and Czachorski [7] and Liu and Perros [14]). Therefore, it is relatively easy to compute the steady-state distribution $\mathbf{v}^*_I(M)$ for each submatrix, $\mathbf{Q}^*_{I\,I}(M)$, I = 1, 2,....,M, and $\mathbf{v}^*_{I'}(M)$ for each submatrix $\mathbf{Q}^*_{I'I'}(M)$, $I' = (M-1)', \ldots, 0'$.

Now, using these steady-state distributions we construct the aggregate matrix $\mathbf{P}(M)$, where each sub-matrix $\mathbf{Q}_{I\,J}(M)$ in the original matrix $\mathbf{Q}(M)$ is represented by a single transition. The upper diagonal elements of $\mathbf{P}(M)$ are given by

$$p_{I\,I+1}(M) = \mathbf{v}_I^{*T}(M) \cdot \mathbf{Q}_{I\,I+1}(M) \cdot \underline{1} \ , \qquad I = 0,1,\ldots,M-1. \tag{11}$$

$$p_{M\,(M-1)'}(M) = \mathbf{v}_M^{*T}(M) \cdot \mathbf{Q}_{M\,M-1}(M) \cdot \underline{1} \tag{12}$$

$$p_{J'J'-1}(M) = \mathbf{v}_{J'}^{*T}(M) \cdot \mathbf{Q}_{J'J'-1}(M) \cdot \underline{1} \ , \quad J' = (M-1)', \ldots, 1'. \tag{13}$$

The lower diagonal elements of $\mathbf{P}(M)$ are given by

$$p_{I\,I-1}(M) = \mathbf{v}_I^{*T}(M) \cdot \mathbf{Q}_{I\,I-1}(M) \cdot \underline{1} \ , \qquad I = 1,2,\ldots,M. \tag{14}$$

$$p_{(M-1)'\,M}(M) = \mathbf{v}_M^{*T}(M) \cdot \mathbf{Q}_{(M-1)'\,M}(M) \cdot \underline{1} \tag{15}$$

$$p_{J'J'+1}(M) = \mathbf{v}_{J'}^{*T}(M) \cdot \mathbf{Q}_{J'J'+1}(M) \cdot \underline{1} \ , \quad J' = (M-2)', \ldots, 0'. \tag{16}$$

The rate matrix $\mathbf{P}(M)$, for M = 2, is as follows:

$$\mathbf{P}(M) = \begin{bmatrix} p_{00}(M) & p_{01}(M) & & & \\ p_{10}(M) & p_{11}(M) & p_{12}(M) & & \\ & p_{21}(M) & p_{22}(M) & p_{21'}(M) & \\ & & p_{1'2}(M) & p_{1'1'}(M) & p_{1'0'}(M) \\ & & & p_{0'1'}(M) & p_{0'0'}(M) \end{bmatrix}, \tag{17}$$

and, in particular, it takes the form

$$\mathbf{P}(M) = \begin{bmatrix} -\mu_3 & \mu_3 & & & \\ \mu_c(1) & -(\mu_c(1)+\mu_3) & \mu_3 & & \\ & \mu_c(2) & -(\mu_c(2)+\mu_3) & \mu_3 & \\ & & \mu_c(2) & -(\mu_c(2)+\mu_3) & \mu_3 \\ & & & \mu_c(2) & -\mu_c(2) \end{bmatrix}, \tag{18}$$

We note that the aggregate rate matrix $\mathbf{P}(M)$ is identical to the rate matrix of a 2-sibling fork/join queue consisting of a composite sibling queue (representing the shorted 2-sibling fork/join queue) and sibling queue 3. The quantity $\mu_c(m)$ is the state-dependent service rate of the composite sibling queue, which is the system's throughput of the shorted 2-sibling fork/join queue with m jobs in it, m = 1,2,...,M.

As in the case of the stationary probability vector $\mathbf{v}^*(M)$ for $\mathbf{Q}^*(M)$, where $\mathbf{v}^*(M) = (\mathbf{v}_0^*(M),..., \mathbf{v}_M^*(M), \mathbf{v}_{(M-1)'}^*(M),..., \mathbf{v}_{0'}^*(M))$, the stationary probability vector $\mathbf{x}(M)$ for $\mathbf{P}(M)$, where $\mathbf{x}(M) = (x_0(M), x_1(M),..., x_M(M), x_{(M-1)'}(M),..., x_{0'}(M))$, can be easily obtained because the closed 2-sibling fork/join queue has a product-form solution. The approximate stationary probability vector $\hat{\mathbf{v}}^{(0)}(M)$ of matrix $\mathbf{Q}(M)$, where $\hat{\mathbf{v}}^{(0)}(M) = (\hat{\mathbf{v}}_0^{(0)}(M),..., \hat{\mathbf{v}}_M^{(0)}(M), \hat{\mathbf{v}}_{(M-1)'}^{(0)}(M),..., \hat{\mathbf{v}}_{0'}^{(0)}(M))$, is as follows,

$$\hat{\mathbf{v}}_I^{(0)}(M) = x_I(M) \cdot \mathbf{v}_I^*(M) , \quad I = 0,1,2,...,M. \tag{19}$$

$$\hat{\mathbf{v}}_{J'}^{(0)}(M) = x_{J'}(M) \cdot \mathbf{v}_{J'}^*(M) , \quad J' = (M-1)',(M-2)', \ldots ,0'. \tag{20}$$

We note that the procedure described above for the 3-sibling fork/join queue, is identical to the procedure described in Liu and Perros [14]. In particular, they first analyzed sibling queues 1 and 2 by shorting sibling queue 3, and obtained the throughput T(i) of the shorted system, for i = 1,2,...,M. Then, they replaced this shorted system by an equivalent composite sibling queue with a state-dependent service rate equal to T(i), and used the composite sibling queue and sibling queue 3 to form a new 2-sibling fork/join queue. The state-dependent service rate of this

composite sibling queue corresponds to the lower diagonal elements of the aggregate matrix $P(M)$. Therefore, we can apply the above procedure rather than using expressions (11)-(16) to obtain the aggregate matrix $P(M)$.

We now apply the Gauss-Seidel iterative method to modify $\hat{\mathbf{v}}^{(0)}(M)$. Let $\hat{\mathbf{v}}^{(1)}(M)$ be the resulting new vector. If $\left|\hat{\mathbf{v}}^{(1)}(M)-\hat{\mathbf{v}}^{(0)}(M)\right| < \epsilon$, then we stop, and $\hat{\mathbf{v}}^{(1)}(M)$ is the stationary probability of $\mathbf{Q}(M)$. Otherwise, set $\mathbf{v}^{*}(M) = \hat{\mathbf{v}}^{(1)}(M)$, and conformally divide $\mathbf{v}^{*}(M)$ into subvectors, so that each subvector corresponds to a block. Normalize each subvector and re-compute the aggregate matrix $\mathbf{P}(M)$, in order to obtain a new stationary probability vector $\mathbf{x}(M)$. Using this vector and $\mathbf{v}^{*}(M)$, calculate $\hat{\mathbf{v}}^{(0)}(M)$. Then, we apply the Gauss-Seidel method to calculate $\hat{\mathbf{v}}^{(1)}(M)$. If $\left|\hat{\mathbf{v}}^{(1)}(M)-\hat{\mathbf{v}}^{(0)}(M)\right| < \epsilon$, then we stop. Else, set $\mathbf{v}^{*}(M) = \hat{\mathbf{v}}^{(1)}(M)$, and repeat the above procedure, until the stationary probability vector of $\mathbf{Q}(M)$ converges.

The procedure described above yields an exact solution for the closed 3-sibling fork/join queue. This procedure, though it was developed independently, turns out to be identical to the algorithm reported by Dodd, McAllister and Stewart [6], Koury, McAllister and Stewart [13], and Cao and Stewart [4], who showed that it yields the exact solution.

3. Approximate Analysis of the K-sibling Fork/Join Queueing System

In this section, we discuss an approximation procedure for analyzing the closed queueing network with a K-sibling fork/join queue shown in figure 1. The approximation procedure makes use of the iterative algorithm presented above. In particular, using the above algorithm, we first reduce the closed K-sibling fork/join queue shown in figure 2 to an approximately equivalent closed 2-sibling fork/join queue. This 2-sibling system is then used to replace the K-sibling system in the original queueing system shown in figure 1. This simplified model is then studied using the procedure described above.

We first analyze the closed K-sibling fork/join queue (see figure 2), where $K \geq 4$, with M customers in it. In particular, let us consider the 3-sibling fork/join queue consisting of sibling queues 1, 2, and 3, obtained by shorting out sibling queues 4 to K. This 3-sibling fork/join queue can be analyzed exactly using the approach described above with m customers in it, where m = 1, 2,...,M. (That is, we use the above procedure M times). The objective of this analysis is not the derivation of the steady-state probability vector of matrix $\mathbf{Q}(m)$, m = 1, 2,...,M. Rather, we are interested in obtaining the steady-state probabilities Pr $(\mathbf{n}_1 | m)$, $\mathbf{n}_1 \epsilon \mathbf{S}_m$, where $\mathbf{n}_1$ is the state of the composite sibling queue c_1 and sibling queue 3 and $\mathbf{S}_m$ is the set of all states for given m, and the rate matrix $\mathbf{P}(m)$, m = 1,2,...,M, which become available upon convergence of the algorithm. As it was mentioned earlier on, $\mathbf{P}(m)$ is the rate matrix of a 2-sibling fork/join queue consisting of a composite sibling queue c_1 and sibling queue 3. The state-dependent service rate of this composite sibling queue c_1 depends on the state of the compo-

site sibling queue and on the state of sibling queue 3. These state-dependent service rates are given on the lower diagonal of the matrix $\mathbf{P}(m)$, m = 1, 2,...,M.

Now, we consider the 4-sibling fork/join queue consisting of sibling queues 1 to 4, with m customers, m = 1, 2,...,M, obtained from the original closed K-sibling fork/join queue by shorting out sibling queues 5 to K. Let (s_1,s_2,s_3,s_4) be the state of the system, where s_i is the number of customers in sibling queue i, i = 1, 2, 3, 4. We analyze this system approximately as a 3-sibling fork/join queue with m customers, m = 1, 2,...,M, consisting of a composite sibling queue c_1, representing sibling queues 1 and 2, sibling queue 3 and sibling queue 4. The state-dependent service rates of the composite sibling queue c_1 are obtained from the lower diagonal of matrix $\mathbf{P}(m)$, calculated from the above analysis of the closed 3-sibling fork/join queue, m = 1, 2,...,M. In an exact decomposition, this service rate should depend on s_4 as well. However, this was not possible to obtain, and this is why our method is approximate. We analyze this 3-sibling fork/join queue using the algorithm described in the previous section, for m = 1, 2,...,M, and obtain the steady-state probabilities Pr $(\mathbf{n}_2|m)$, $\mathbf{n}_2 \epsilon \mathbf{S}_m$, where $\mathbf{n}_2$ is the state of the composite sibling queue c_2 and sibling queue 4, and $\mathbf{S}_m$ is the set of all states for given m, and the rate matrix $\mathbf{P}(m)$, m = 1,2,...,M.

Now, let us consider the 5-sibling fork/join queue consisting of sibling queues 1 to 5, with m customers, m = 1, 2,...,M. This is analyzed as a 3-sibling fork/join queue consisting of a composite sibling queue c_2, representing sibling queues 1, 2, and 3, sibling queue 4 and sibling queue 5, with m customers, m = 1, 2,...,M. As

described above, the service rates of the composite sibling queue are obtained from the lower diagonal of matrix $\mathbf{P}(m)$, calculated from the above analysis of the 4-sibling fork/join queue. We analyze this 3-sibling fork/join queue for m = 1, 2,...,M, and obtain the steady-state probabilities Pr $(\mathbf{n}_3|m)$, $\mathbf{n}_3 \epsilon \mathbf{S}_m$, where $\mathbf{n}_3$ is the state of the composite sibling queue c_3 and sibling queue 5, and $\mathbf{S}_m$ is the set of all states for given m, and the rate matrix $\mathbf{P}(m)$, m = 1,2,...,M.

We proceed in this fashion until we analyze a 3-sibling fork/join queue consisting of a composite sibling queue c_{K-3}, representing sibling queues 1, 2,..., and K-2, and sibling queue K-1 and sibling queue K, with m customers, m = 1, 2,...,M. Again, of interest is the steady-state probabilities Pr $(\mathbf{n}_{K-2}|m)$, $\mathbf{n}_{K-2} \epsilon \mathbf{S}_m$, where $\mathbf{n}_{K-2}$ is the state of the composite sibling queue c_{K-2} and sibling queue K, and $\mathbf{S}_m$ is the set of all states for given m, and the rate matrix $\mathbf{P}(m)$, m = 1, 2,...M. Using the lower diagonal elements of this matrix, we reduce the closed 3-sibling fork/join queue to a closed 2-sibling fork/join queue, consisting of a composite sibling queue c_{K-2} and sibling queue K. Thus, the original queueing system, shown in figure 1, can be approximately reduced to a model consisting of the fork node and a 2-sibling fork/join queue, as shown in figure 3, with M customers. Now, it is easy to see that this model can be analyzed exactly using the approach developed in section 2. Thus, as a last step, we analyze the system shown in figure 3 with M customers.

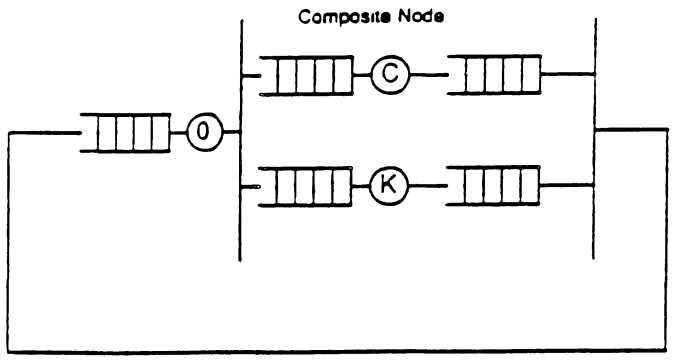

figure 3 : The reduced fork/join model

The above procedure yields the system throughput and the marginal queue-length probability distribution for the fork node, the *Kth* sibling queue and composite sibling queue c_{K-2}. Performance measures of the other sibling queues can be obtained by working backwards. In particular, using the marginal queue-length probability of the c_{K-2} composite sibling queue $Pr_{c_{K-2}}$ (m), m = 1, 2,...,M, we have

$$Pr(\mathbf{n}_{K-3}) = Pr(\mathbf{n}_{K-3}|m) \cdot Pr_{c_{K-2}}(m), \quad m = 1,2,...,M, \tag{21}$$

where $Pr(\mathbf{n}_{K-3})$ is the joint probability distribution of the composite sibling queue c_{K-3} and sibling queue K-1. Using $Pr(\mathbf{n}_{K-3})$, the marginal queue-length probability distribution of the sibling queue K-1 and the composite sibling queue c_{K-3} can be easily obtained.

We can proceed backwards in this fashion until we obtain the marginal queue-lehgth probability distribution for sibling queues 1 and 2. Thus, the mean

queue length of each sibling queue can be easily obtained. Consequently, the mean queue length of each synchronization queue can also be obtained.

4. Results Analysis

The above algorithm was implemented on a VAX-11/785 to analyze the fork/join model shown in figure 1 with 3, 4 and 8 siblings. For 3 and 4 siblings, the approximate results were compared against exact numerical values, and for 8 siblings they were compared against simulation results. The results for both homogeneous cases and non-homogeneous cases are presented in tables 3 to 8 for the test case shown in table 2. Each table gives the approximate and exact (or simulation) results for the system throughput, mean queue lengths, and the mean response time of the fork/join operation for various values of μ_0. The relative error ([(exact (or simulation) - approximate)/exact (or simulation)] 100%) is also given.

It was observed that the approximation procedure gives very good results for the system throughput (the relative error is less than 3%), the mean response time of the fork/join operation (the relative error is less than 5% on the average) and the mean queue length for the fork node and the mean queue length for each sibling queue (the relative error is less than 5% on the average). However, it gives results which are an upper bound of the mean response time of the fork/join operation and a lower bound of the system throughput, for both homogeneous and non-homogeneous cases.

Case 1 :	3-sibling homogeneous fork/join model. $\mu_1 = \mu_2 = \mu_3 = 2$. M = 10. See table 3.
Case 2 :	3-sibling non-homogeneous fork/join model. $\mu_1 = 1$, $\mu_2 = 2$, $\mu_3 = 3$, M = 10. See table 4.
Case 3 :	4-sibling homogeneous fork/join model, $\mu_1 = \mu_2 = \mu_3 = \mu_4 = 1$, M = 5. See table 5.
Case 4 :	4-sibling non-homogeneous fork/join model, $\mu_1 = 1$, $\mu_2 = 2$, $\mu_3 = 3$, $\mu_4 = 4$, M = 5. See table 6.
Case 5 :	8-sibling homogeneous fork/join model, $\mu_1 = \mu_2 = ... = \mu_8 = 1$, M = 15. See table 7.
Case 6 :	8-sibling non-homogeneous fork/join model, $\mu_1 = \mu_2 = 1$, $\mu_3 = \mu_4 = 2$, $\mu_5 = \mu_6 = 3$, $\mu_7 = \mu_8 = 4$, M = 15. See table 8.

table 2 : Validation test cases

The approximation procedure is, in general, very fast. The efficiency of the approximation procedure can be judged by comparing its CPU time against that of the exact numerical procedure. For instance, for the 3-sibling fork/join model with M = 10 jobs, the average CPU time of the exact numerical procedure was 179.4 seconds, whereas the average CPU time of the approximation procedure for the homogeneous model was 68.46 seconds, for the non-homogeneous model was 37.16 seconds. For the 4-sibling fork/join model with M = 5 jobs, the average CPU time of the exact numerical procedure was 112.3 seconds, and the average

CPU time of the approximation procedure for the homogeneous model was 16.36 seconds, for the non-homogeneous model was 11.16 seconds. This procedure gives more accurate results than the one presented by Liu and Perros [14]. This is because, we have obtained the state-dependent service rate of a composite sibling queue so that not only it depends on the state of composite sibling queue, but also it depends on the state of a third sibling queue.

5. Conclusions

We developed an approximation procedure for analyzing a closed queueing network with a K-sibling fork/join queue. The approximation procedure is based on nearly complete decomposability and Gauss-Seidel iterative method. It was shown through a number of examples that it gives results which are a lower bound of the system throughput and an upper bound of the mean response time of the fork/join operation.

We note that the original queueing network shown in figure 1 can be also approximately reduced to a two-node closed queueing networks consisting of the original fork node and a composite node with a state-dependent service rate equal to the approximate throughput of the closed K-sibling fork/join queue. This closed queueing network can be easily analyzed. Then we can apply the same backward approach to obtain the marginal queue length probability distribution for each sibling queue. The performance measures obtained from this approach are very close to the approximate results obtained from section 3.

Finally, we note that in the paper by Liu and Perros [14], they analyzed the same problem using decomposition and aggregation. In particular, the closed K-sibling fork/join queue was reduced iteratively to an approximately equivalent closed 2-sibling fork/join queue. The reduction was carried out by successively replacing closed 2-sibling fork/join queues by a composite sibling queue. The state-dependent service rate of the composite node was set equal to the system throughput of the 2-sibling fork/join queue. However, due to the interdependency between siblings, the state-dependent service rate of the composite node not only depends on the state of the 2-sibling fork/join queue, but also depends on the state of the other sibling queues in the K-sibling fork/join queue. In this paper, we substitute a 2-sibling fork/join queue by a composite sibling queue, whose state-dependent service rate not only depends on the state of the 2-sibling fork/join queue, but also it depends on the state of a third sibling queue.

REFERENCES

[1] **Baccelli, F. and Makowski, A.M.**, "Simple Computable Bounds for the Fork-Join Queue", Proceeding of the 19*th* Annual Conference on Information Science and Systems, The John Hopkins University, Baltimore, Maryland, March 1985, pp. 436-441.

[2] **Baccelli, F., Makowski, A.M. and Shwartz, A.**, "The Fork-Join Queue and Related Systems with Synchronization Constraints : Stochastic Ordering, Approximations and Computable Bounds ", Electrical Engineering Technical Report, University of Maryland, College Park, January 1987.

[3] **Baccelli, F. and Massey, W.A. and Towsley, D.**, "Acyclic Fork-Join Queueing Networks", Internal Report, Computer Science Department, University of Massachusetts, April 1987. Systems and Their Performance, 1987

[4] **Cao W. L. and Stewart, W.J.**, "Iterative Aggregation/Disaggregation Techniques for Nearly Uncoupled Markov Chains", J. ACM, vol. 32, No. 3, July 1985, 702-719.

[5] **Courtois, P.J.**, "Decomposability: Queueing and Computer System Applications", Academic Press, 1977.

[6] **Dodd, S.L., McAllister D.F. and Stewart W.J.**, "An Iterative Method for the Exact Solution of Coxian Queueing Networks", Proceedings ACM/SIGMETRICS Conf. on Measurement and Modeling of Computer Systems, September 1981, 97-104.

[7] **Duda, A. and Czachorski, T.**, "Performance Evaluation of Fork and Join Synchronization Primitives", to appear in ACTA Information.

[8] **Duda, A.**, "Approximate Performance Analysis of parallel Systems", Proc. of 2nd Inf. Workshop on Appl. Mathematics and Performance/Reliability Models of Computer/Communication Systems, Rome, Italy, May 25-27, 1987.

[9] **Flatto,L. and Hahn, S.**, "Two Parallel Queues Created by Arrivals with Two Demands I", SIAM J. Appl. Math. vol. 44, Oct. 1984, 1041-1053.

[10] **Flatto,L.**, "Two Parallel Queues Created by Arrivals with Two Demands I", SIAM J. Appl. Math. vol. 45, Oct. 1985, 861-878.

[11] **Heidelberger, P. and Trivedi, S.K.**, "Queueing Network Models for Parallel Processing with Asynchronous Tasks", IEEE Trans. on Comp. vol. 31, Nov. 1982, 1099-1109.

[12] **Heidelberger, P. and Trivedi, S.K.**, "Analytic Queueing Models for Programs with Internal Concurrency", IEEE Trans. on Comp. vol. 32, Jan. 1983, 73-82.

[13] **Koury, R., McAllister, D.F. and Stewart, W.J.**, "Methods for Computing Stationary Distributions of Nearly Completely Decomposable Markov

Chains", SIAM J. Disc. Math. 5,2 (1984), 164-186.

[14] **Liu, Y.C. and Perros, H.G.**, "Approximate Analysis of a Closed Fork/Join Model", Technical Report - 88, CCSP, ECE, NCSU.

[15] **Mailles, D.**, "Files d'Attente Descriptives Pour la Modelisation de la Synchronisation dans des Systemes Informatiques", These de Doctorate D'Etat, Univ. Paris 6, 1987.

[16] **Nelson, R. and Tantawi, A.N.**, "Approximate Analysis of Fork/Join Synchronization in Parallel Queues", IBM Research Report RC 11481.

Table 3 : 3-sibling homogeneous fork/join model: M =10: $\mu_1=\mu_2=\mu_3=2$

System Throughput

μ_0/μ_1	Exact	Approx.	Error
10.0	1.853	1.853	0.00%
5.0	1.852	1.851	0.04%
1.0	1.712	1.708	0.26%
0.5	0.998	0.998	0.00%
0.1	0.200	0.200	0.00%

Mean Response Time of Fork/Join Operation

μ_0/μ_1	Exact	Approx.	Error
10.0	5.342	5.341	0.02%
5.0	5.279	5.280	0.02%
1.0	3.751	3.763	0.32%
0.5	1.705	1.731	1.52%
0.1	1.005	1.008	0.34%

Mean Queue Length at Fork node

μ_0/μ_1	Exact	Approx.	Error
10.0	0.1014	0.1016	0.17%
5.0	0.2238	0.2247	0.38%
1.0	3.5780	3.5745	0.10%
0.5	8.2980	8.2725	0.31%
0.1	9.7990	9.7983	0.01%

Mean Queue Length at Sibling Queue 1

μ_0/μ_1	Exact	Approx.	Error
10.0	6.203	6.214	0.18%
5.0	6.123	6.133	0.16%
1.0	3.934	3.892	1.06%
0.5	0.991	0.991	0.00%
0.1	0.111	0.111	0.00%

Mean Queue Length at Synchronization Queue 1

μ_0/μ_1	Exact	Approx.	Error
10.0	3.69600	3.6845	0.31%
5.0	3.65300	3.6424	0.29%
1.0	2.48800	2.5332	1.82%
0.5	0.71020	0.7369	3.76%
0.1	0.08994	0.0906	0.72%

Table 4 : 3-Sibling non-homogeneous fork/join model: M =10: $\mu_1 = 1$, $\mu_2 = 2$, $\mu_3 = 3$.

System Throughput

μ_0/μ_1	Exact	Approx.	Error
10.0	0.9998	1.0000	0.02%
5.0	0.9998	1.0000	0.02%
1.0	0.9090	0.9071	0.21%
0.5	0.4997	0.4997	0.00%
0.1	0.1000	0.1000	0.00%

Mean Response Time of Fork/Join Operation

μ_0/μ_1	Exact	Approx.	Error
10.0	9.8910	9.8890	0.02%
5.0	9.7520	9.7501	0.02%
1.0	5.5620	5.6328	1.27%
0.5	2.1610	2.2155	2.52%
0.1	1.3240	1.3300	0.45%

Mean queue length at Fork Node

μ_0/μ_1	Exact	Approx.	Error
10.0	0.1111	0.1111	0.00%
5.0	0.2499	0.2499	0.00%
1.0	4.9440	4.8905	1.08%
0.5	8.9200	8.8929	0.30%
0.1	9.8680	9.8670	0.01%

Mean Queue Length at Sibling Queue 1

μ_0/μ_1	Exact	Approx.	Error
10.0	9.8880	9.8880	0.00%
5.0	9.7490	9.7490	0.00%
1.0	4.9980	5.0590	1.22%
0.5	0.9945	1.0270	3.26%
0.1	0.1111	0.1120	0.80%

Mean Queue Length at Synchronization Queue 1

μ_0/μ_1	Exact	Approx.	Error
10.0	0.001106	0.0009	18.60%
5.0	0.001311	0.0011	16.10%
1.0	0.057850	0.0505	12.70%
0.5	0.085330	0.0801	6.13%
0.1	0.021140	0.0219	3.60%

Mean Queue Length at Sibling Queue 2

μ_0/μ_1	Exact	Approx.	Error
10.0	0.99880	0.9989	0.01%
5.0	0.99870	0.9987	0.00%
1.0	0.81910	0.8208	0.20%
0.5	0.33300	0.3375	1.35%
0.1	0.05263	0.0528	0.42%

Mean Queue Length at Synchronization Queue 2

μ_0/μ_1	Exact	Approx.	Error
10.0	8.89000	8.8900	0.00%
5.0	8.75100	8.7514	0.00%
1.0	4.23700	4.2887	1.22%
0.5	0.74690	0.7696	3.03%
0.1	0.07962	0.0801	0.55%

Mean Queue Length at Sibling Queue 3

μ_0/μ_1	Exact	Approx.	Error
10.0	0.49970	0.4998	0.02%
5.0	0.49970	0.4998	0.02%
1.0	0.43180	0.4304	0.32%
0.5	0.19980	0.1998	0.00%
0.1	0.03448	0.0345	0.00%

Mean Queue Length at Synchronization Queue 3

μ_0/μ_1	Exact	Approx.	Error
10.0	9.38900	9.3890	0.00%
5.0	9.25000	9.2502	0.00%
1.0	4.62400	4.6791	1.19%
0.5	0.88000	0.9072	3.08%
0.1	0.09777	0.0985	0.74%

Table 5 : 4-sibling homogeneous fork/join model; M =5; $\mu_1=\mu_2=\mu_3=\mu_4=1$

System Throughput

μ_0/μ_1	Exact	Approx.	Error
10.0	0.8315	0.8281	0.41%
5.0	0.8288	0.8242	0.56%
1.0	0.7244	0.7172	0.99%
0.5	0.4786	0.4765	0.44%
0.1	0.1000	0.1000	0.00%

Mean Response Time of Fork/Join Operation

μ_0/μ_1	Exact	Approx.	Error
10.0	5.9050	5.9302	0.43%
5.0	5.8020	5.8330	0.53%
1.0	4.6400	4.6954	1.19%
0.5	3.4300	3.5100	2.30%
0.1	2.2760	2.3170	1.80%

Mean Queue Length at Fork Node

μ_0/μ_1	Exact	Approx.	Error
10.0	0.08913	0.08924	0.12%
5.0	0.19150	0.19246	0.50%
1.0	1.63900	1.63246	0.40%
0.5	3.35900	3.32780	0.93%
0.1	4.77200	4.76830	0.08%

Mean Queue Length at Sibling Queue 1

μ_0/μ_1	Exact	Approx.	Error
10.0	2.69100	2.6936	0.10%
5.0	2.63000	2.6285	0.06%
1.0	1.79200	1.7603	1.77%
0.5	0.84750	0.8400	0.88%
0.1	0.11100	0.1110	0.00%

Mean Queue Length at Synchronization Queue 1

μ_0/μ_1	Exact	Approx.	Error
10.0	2.21900	2.2172	0.08%
5.0	2.17900	2.1791	0.00%
1.0	1.56900	1.6072	2.44%
0.5	0.79390	0.8322	4.82%
0.1	0.11660	0.1206	3.42%

Table 6 : 4-sibling non-homogeneous fork/join model: M =5; $\mu_1 = 1$, $\mu_2 = 2$, $\mu_3 = 3$, $\mu_4 = 4$.

System Throughput

μ_0/μ_1	Exact	Approx.	Error
10.0	0.9997	0.9992	0.05%
5.0	0.9992	0.9983	0.09%
1.0	0.8317	0.8252	0.78%
0.5	0.4919	0.4910	0.18%
0.1	0.1000	0.1000	0.00%

Mean Response Time of Fork/Join Operation

μ_0/μ_1	Exact	Approx.	Error
10.0	4.8905	4.8930	0.05%
5.0	4.7572	4.7596	0.05%
1.0	3.1121	3.1683	1.80%
0.5	2.0248	2.0839	2.92%
0.1	1.3400	1.3570	1.27%

Mean Queue Length at Fork Node

μ_0/μ_1	Exact	Approx.	Error
10.0	0.1106	0.1109	0.27%
5.0	0.2470	0.2485	0.60%
1.0	2.4120	2.3855	1.10%
0.5	4.0040	3.9768	0.68%
0.1	4.8660	4.8643	0.03%

Mean Queue Length at Sibling Queue 1

μ_0/μ_1	Exact	Approx.	Error
10.0	4.8560	4.8560	0.00%
5.0	4.7150	4.7140	0.02%
1.0	2.4840	2.5210	1.46%
0.5	0.9037	0.9346	3.42%
0.1	0.1111	0.1124	1.17%

Mean Queue Length at Synchronization Queue 1

μ_0/μ_1	Exact	Approx.	Error
10.0	0.0330	0.0331	0.30%
5.0	0.0384	0.0375	2.34%
1.0	0.1043	0.0935	10.35%
0.5	0.0923	0.0886	4.00%
0.1	0.0229	0.0233	1.75%

Mean Queue Length at Sibling Queue 2

μ_0/μ_1	Exact	Approx.	Error
10.0	0.9620	0.9622	0.02%
5.0	0.9561	0.9549	0.13%
1.0	0.6714	0.6710	0.06%
0.5	0.3229	0.3274	1.39%
0.1	0.0526	0.0529	0.63%

Mean Queue Length at Synchronization Queue 2

μ_0/μ_1	Exact	Approx.	Error
10.0	3.9270	3.9269	0.00%
5.0	3.7970	3.7966	0.01%
1.0	1.9170	1.9435	1.38%
0.5	0.6731	0.6958	3.37%
0.1	0.0814	0.0828	1.68%

Mean Queue Length at Sibling Queue 3

μ_0/μ_1	Exact	Approx.	Error
10.0	0.4925	0.4922	0.06%
5.0	0.4913	0.4901	0.24%
1.0	0.3737	0.3686	1.36%
0.5	0.1951	0.1947	0.20%
0.1	0.0345	0.0345	0.00%

Mean Queue length at Synchronization Queue 3

μ_0/μ_1	Exact	Approx.	Error
10.0	4.3970	4.3969	0.00%
5.0	4.2620	4.2614	0.01%
1.0	2.2140	2.2459	1.44%
0.5	0.8009	0.8285	3.45%
0.1	0.0996	0.1012	1.60%

Mean Queue Length at Sibling Queue 4

μ_0/μ_1	Exact	Approx.	Error
10.0	0.3298	0.3312	0.42%
5.0	0.3294	0.3305	0.33%
1.0	0.2583	0.2558	0.99%
0.5	0.1398	0.1394	0.29%
0.1	0.0256	0.0256	0.00%

Mean Queue Length at Synchronization Queue 4

μ_0/μ_1	Exact	Approx.	Error

10.0	4.5600	4.5579	0.05%
5.0	4.4240	4.4210	0.07%
1.0	2.3300	2.3587	1.23%
0.5	0.8563	0.8837	3.20%
0.1	0.1084	0.1101	1.56%

Table 7 : 8-sibling homogeneous fork/join model: $M = 15$: $\mu_1 = \mu_2 = \ldots = \mu_8 = 1$

System Throughput

μ_0/μ_1	Simulation	Approx.	Error
10.0	0.9187	0.9036	1.64%
5.0	0.9132	0.9028	1.14%
1.0	0.8659	0.8546	1.30%
0.5	0.4931	0.4999	1.38%
0.1	0.0976	0.1000	2.46%

Mean Response Time of Fork/Join Operation

μ_0/μ_1	Simulation	Approx.	Error
10.0	16.3130	16.4910	1.09%
5.0	16.2493	16.3720	0.76%
1.0	12.2200	12.7270	4.15%
0.5	4.8097	5.3490	11.21%
0.1	2.9668	3.2340	9.01%

Mean Queue Length at Fork Node

μ_0/μ_1	Simulation	Approx.	Error
10.0	0.10210	0.09904	3.00%
5.0	0.22056	0.21898	0.72%
1.0	4.40822	4.11827	6.58%
0.5	12.63000	12.32620	2.41%
0.1	14.70000	14.67656	0.16%

Mean Queue Length at Sibling Queue 1 :

μ_0/μ_1	Simulation	Approx.	Error
10.0	6.8389	6.7605	1.15%
5.0	6.6968	6.7015	0.07%
1.0	4.7367	4.6149	2.57%
0.5	0.9742	0.9989	2.54%
0.1	0.1129	0.1111	1.62%

Mean Queue Length at Synchronization Queue 1 :

μ_0/μ_1	Simulation	Approx.	Error
10.0	8.0567	8.1405	1.04%
5.0	8.0825	8.0795	0.04%
1.0	5.8550	6.2623	6.95%
0.5	1.3956	1.6749	20.01%
0.1	0.1875	0.2123	13.24%

Table 8 : 8-sibling non-homogeneous fork/join model. $\mu_1 = \mu_2 = 1$, $\mu_3 = \mu_4 = 2$, $\mu_5 = \mu_6 = 3$, $\mu_7 = \mu_8 = 1$, $M = 15$.

System Throughput

μ_0/μ_1	Simulation	Approx.	Error
10.0	0.9770	0.9675	0.97%
5.0	0.9668	0.9672	0.04%
1.0	0.9171	0.9086	0.93%
0.5	0.5034	0.5000	0.70%
0.1	0.0986	0.1000	1.40%

Mean Response Time of Fork/Join Operation

μ_0/μ_1	Simulation	Approx.	Error
10.0	15.40820	15.3932	0.10%
5.0	15.10850	15.2610	1.00%
1.0	9.95020	9.9587	0.10%
0.5	3.10450	3.2280	3.98%
0.1	1.81640	1.9400	6.80%

Mean Queue Length at Fork Node

μ_0/μ_1	Simulation	Approx.	Error
10.0	0.10544	0.1071	1.57%
5.0	0.24202	0.2396	1.00%
1.0	6.32950	5.9515	5.97%
0.5	13.43600	13.3860	0.37%
0.1	14.82000	14.8060	0.10%

Mean Queue Length at Sibling Queue 1

μ_0/μ_1	Simulation	Approx.	Error
10.0	11.0072	11.0500	0.39%
5.0	10.9437	10.9500	0.06%
1.0	6.3467	6.6660	4.94%
0.5	1.0543	1.1050	4.80%
0.1	0.1083	0.1177	8.68%

Mean Queue Length at Synchronization queue 1

μ_0/μ_1	Simulation	Approx.	Error
10.0	3.8873	3.8429	1.14%
5.0	3.8142	3.8104	0.10%
1.0	2.3237	2.3885	2.79%
0.5	0.5092	0.5090	0.04%
0.1	0.0708	0.0763	7.77%

Mean Queue Length at Sibling Queue 3

μ_0/μ_1	Simulation	Approx.	Error
10.0	0.90400	0.9174	1.48%
5.0	0.92257	0.9167	0.64%
1.0	0.85246	0.8247	3.26%
0.5	0.33888	0.3392	0.09%
0.1	0.05314	0.5252	1.17%

Mean Queue Length at Synchronization Queue 3

μ_0/μ_1	Simulation	Approx.	Error
10.0	13.9905	13.9755	0.11%
5.0	13.8354	13.8437	0.06%
1.0	7.8179	8.2238	5.19%
0.5	1.2245	1.2748	4.11%
0.1	0.1259	0.1415	12.39%

Mean Queue Length at Sibling Queue 5

μ_0/μ_1	Simulation	Approx.	Error
10.0	0.47130	0.4747	0.73%
5.0	0.47450	0.4745	0.00%
1.0	0.43900	0.4333	1.30%
0.5	0.20200	0.2014	0.30%
0.1	0.03369	0.0343	1.78%

Mean Queue Length at Synchronization Queue 5

μ_0/μ_1	Simulation	Approx.	Error
10.0	14.4233	14.4182	0.03%
5.0	14.2835	14.2859	0.02%
1.0	8.2314	8.6152	4.66%
0.5	1.3614	1.4126	3.76%
0.1	0.1454	0.1597	9.84%

Mean Queue Length at Sibling Queue 7

μ_0/μ_1	Simulation	Approx.	Error
10.0	0.319318	0.3189	0.13%
5.0	0.322000	0.3188	1.00%
1.0	0.297300	0.2942	1.04%
0.5	0.143550	0.1441	0.38%
0.1	0.025460	0.0255	0.16%

Mean Queue Length at Synchronization Queue 7

μ_0/μ_1	Simulation	Approx.	Error
10.0	14.5752	14.5746	0.01%
5.0	14.4368	14.4416	0.03%
1.0	8.3730	8.7543	4.55%
0.5	1.4199	1.4699	3.52%
0.1	0.1536	0.1685	9.70%