Approximate Analysis of a Closed Fork/Join Model

Y.C. Liu
and
H.G. Perros

Center for Communications and Signal Processing
Department of Computer Science
North Carolina State University

CCSP-TR-88/15

April 1988

Abstract

An approximation algorithm for analyzing a closed queueing systems with a K-sibling fork/join queue is presented. The procedure is based on decomposition and aggregation. 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. A modification of this procedure, applicable only to the homogeneous case, is also presented. This procedure was found to give very good results for the system throughput and the mean response time of the fork/join operation (the relative error is less than 3%).

1. Introduction

In recent years, there has been a growing 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 typesof operation are known as fork/join, or disassembly/assembly in production systems. 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 [6,7] 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 [12] 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 [11] 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 [12]. 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 [8] 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 [9], 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 [4]. The numerical approach is inherently limited to small problems. Also, Duda [5] developed an approximation algorithm for analyzing a series-parallel fork/join networks by constructing an approximately equivalent queueing network with a product-form solution.

In this paper, we consider a closed queueing system with a K-sibling fork/join queue ($K \geq 2$). The fork/join queue may consist of homogeneous or non-homogeneous exponential servers. The buffer size of each queue is infinite. Using the decomposition and aggregation approach, we iteratively reduce the K-sibling fork/join queue into a 2-sibling fork/join queue. This 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. For the homogeneous case, we empirically observed that the difference of the system throughput between the approximate solution and the exact solution proportionally increases as the number of siblings increases. Hence, we developed a modification procedure in order to improve the accuracy of the approximation algorithm. The relative error of the system throughput and the response time of the modified approximation algorithm was found to be less then 3%.

In the following section, we describe the fork/join model. In section 3, we present an approximation algorithm based on decomposition and aggregation. In section 4, we describe the modification procedure. Finally, the conclusions are given in section 5.

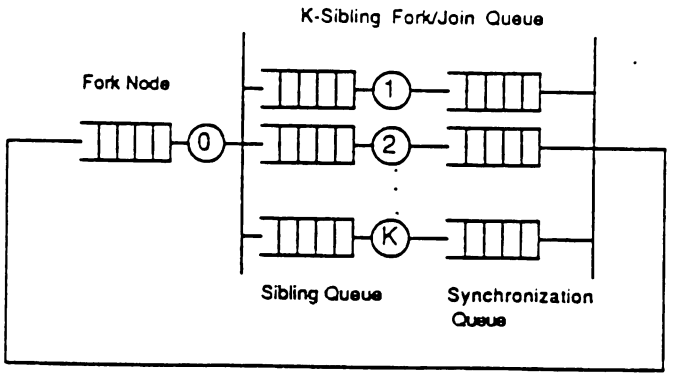

figure 1: The System Under Study

2. Model Description

The model studied in this paper is a closed queueing network consisting of a fork node and a K-sibling fork/join queue, as shown in figure 1. The K-sibling fork/join queue consists of K servers arranged in parallel ($K \geq 2$). Each server has its own queue, hereafter called the sibling queue. For each sibling queue there is a synchronization queue. The capacity of each queue is assumed to be infinite, the service time of each server is exponentially distributed, and the service discipline is FIFO. Let μ_i , i = 1, 2,...,K, be the service rate at sibling queue i, and let μ_0 be the service rate at the fork node. The fork/join queue is referred to as homogeneous if $\mu_i = \mu$, i = 1, 2,...,K. Let M be the number of jobs in the system. Each job is split up into K siblings (fork operation) upon completion of its service at the fork node. Each sibling joins a different sibling queue. Upon service completion, a sibling enters its synchronization queue where it waits for the other

siblings associated with the same job. When all siblings have completed their service, i.e., each sibling is in its synchronization queue, they recompose immediately to the original job which joins the fork node. At time t, let P(t) be the number of jobs in the fork node, $S_i(t)$ the number of siblings in sibling queue i, $W_i(t)$ the number of siblings that have been serviced and are waiting in the ith synchronization queue, i = 1,2,...,K. Then, at any given time t, M = P(t) + $S_i(t)$ + $W_i(t)$, i = 1,2,...,K.

The motivation behind studying this fork/join system as a closed queueing network, is that it allows us to incorporate approximately fork/join operations in BCMP type of queueing networks. Let us consider, for example, a queueing network of the BCMP type, and let us assume that jobs after completing service at node i, are split up into K siblings. Each sibling enters a different queue. When all siblings complete their service, they merge into a single job which joins some other queue in the system. The BCMP queueing network obtained by shorting out the fork/join queue can be replaced approximately by an equivalent server, so that the resulting queueing network is the one shown in figure 1. This approximate method was not investigated, seeing that such an approximation is common and quite well understood (cf. Perros, Nilsson and Liu [13]).

3. The Approximation Procedure

In this section, we discuss an approximation procedure for analyzing the closed queueing network with a K-sibling fork/join queue described above and

shown in figure 1.

We developed both a numerical procedure and an approximation algorithm. The numerical procedure involved the following steps: (1) generation of all the states of the system; (2) generation of the rate matrix Q; and (3) solution of the linear system $Qx^T = 0$ in order to obtain the steady state probability distribution. We note that the state space increases very fast, seeing that the number of states is equal to $(M+1)^K$. In view of this, the numerical procedure is only suitable for analyzing small systems. The approximation procedure is based on decomposition and aggregation, and it can be used for any number of siblings.

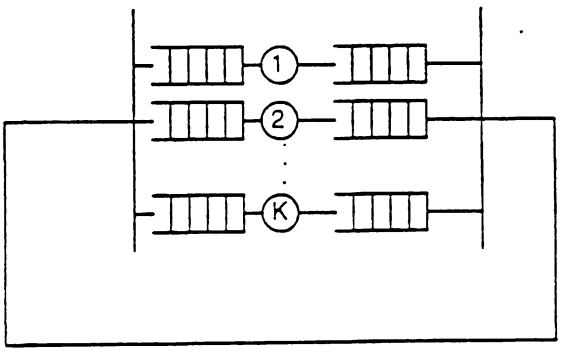

figure 2: The K-Sibling Fork/Join Queue as a Closed Queueing Network

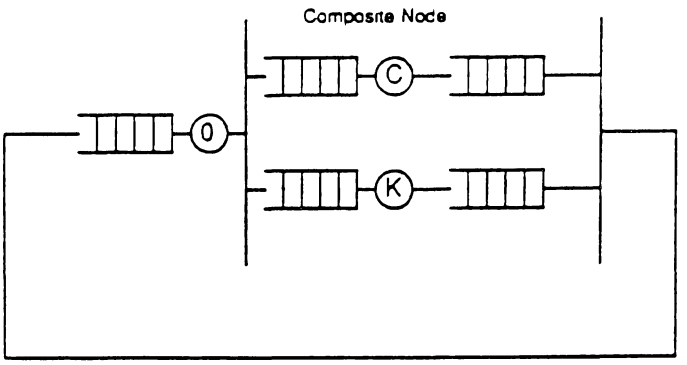

figure 3: The Reduced Fork/Join Model

The approximation procedure can be briefly summarized as follows. We first analyze approximately the K-sibling fork/join queue by shorting out the fork node, i.e., analyze the K-sibling fork/join queue as a closed queueing network independently of the fork node, as shown in figure 2. In particular, using decomposition and aggregation, this system is approximately reduced to a 2-sibling fork/join queue, which is then used to replace the K-sibling fork/join queue in the original queueing network as shown in figure 3. This queueing system is then analyzed numerically, using the numerical procedure described above.

The study of the K-sibling fork/join queue as an independent subnetwork was motivated by the fact that the average rate of interaction within the fork/join queue is much higher than the average rate of interaction between the fork node and the fork/join queue (see figure 4). This is primarily due to the synchronization among the siblings. In view of this, the fork node and the K-sibling fork/join queue can be seen as being "weakly coupled".

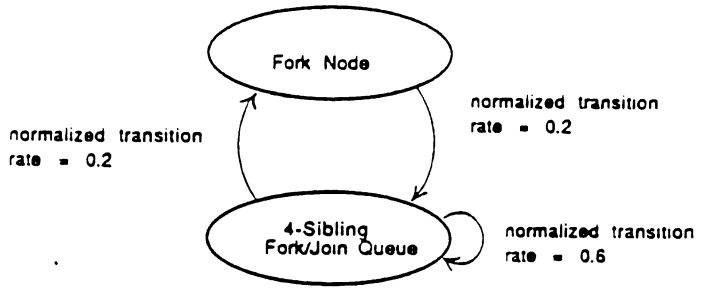

figure 4: Normalized Transition Rates Between the Fork Node and a 4-Sibling Homogeneous Fork/Join Queue with M = 2

Let us first consider the fork/join queue shown in figure 2, assuming that K = 2. Let $\mu_1(i)$ and $\mu_2(j)$ be the state-dependent service rate of sibling queue 1 and 2, i,j = 1,2,...,M. Let $P_{i,j}$ be the steady state probability that there are i and j jobs in sibling queue 1 and 2 respectively. It can be easily shown that this system has a product-form solution (see also Duda and Czachorski [4]). The stationary equations are as follows :

$$P_{M,M}\ (\mu_1(M)+\mu_2(M)) = \mu_1(M)\ P_{M,M-1} + \mu_2(M)\ P_{M-1,M} \tag{1}$$

$$P_{M,N}\ (\mu_1(M)+\mu_2(N)) = \mu_1(M)\ P_{M,N-1} + \mu_2(N+1)\ P_{M,N+1} \tag{2}$$

$$1 \leq N \leq M-1$$

$$P_{M,0}\ \mu_1(M) = P_{M,1}\ \mu_2(1) \tag{3}$$

$$P_{N,M}\ (\mu_1(N)+\mu_2(M)) = \mu_1(N+1)\ P_{N+1,M} + \mu_2(M)\ P_{N-1,M} \tag{4}$$

$$1 \leq N \leq M-1$$

$$P_{0,M}\ \mu_2(M) = P_{1,M}\ \mu_1(1) \tag{5}$$

$$P_{M,M} + \sum_{i=0}^{M-1} P_{M,i} + \sum_{j=0}^{M-1} P_{j,M} = 1 \tag{6}$$

We can easily obtain

$$P_{N,M} = \prod_{j=N+1}^{M} \rho_2(j) \; P_{M,M} \quad 0 \le N \le M-1 \tag{7}$$

$$P_{M,N} = \prod_{j=N+1}^{M} \rho_1(j) \; P_{M,M} \quad 0 \le N \le M-1 \tag{8}$$

and

$$P_{M,M} = \frac{1}{1 + \sum_{i=0}^{M-1} \prod_{j=i+1}^{M} \rho_2(j) + \sum_{j=0}^{M-1} \prod_{N=j+1}^{M} \rho_1(N)} \tag{9}$$

where

$$\rho_2(j) = \frac{\mu_1(j)}{\mu_2(M)} \quad \text{and} \quad \rho_1(j) = \frac{\mu_2(j)}{\mu_1(M)}$$

The system throughput, T(M), is :

$$T(M) = \sum_{i=1}^{M} \mu_1(i) \; P_{i,M} + \sum_{j=0}^{M-1} \mu_1(M) \; P_{M,j} \tag{10}$$

Using expression (10) in conjunction with decomposition and aggregation, we can now reduce approximately the K-sibling fork/join queue to a 2-sibling fork/join queue. Let us first number the sibling queues in figure 2 from 1 to K starting from the top. The approximation algorithm can now be summarized as follows:

Step 1. Consider the closed queueing system consisting of sibling queues 1 and 2 (and their corresponding synchronization queues) obtained from figure 2 by shorting out the remaining sibling queues. Let T (i), be the throughput of this system, obtained using expression (10), when there are i customers in it, i = 1,2,...,M. Replace this system by an equivalent composite node, call it C_1, with a state-dependent service rate equal to T (i). Now, use composite node C_1 and sibling queue 3 to form a new 2-sibling fork/join queue. As above, construct an equivalent composite node, call it C_2. Proceed as above until the K-1 sibling queues are replaced by an equivalent composite node C_{K-2}. The K-sibling fork/join queue has now been reduced to a 2-sibling fork/join queue consisting of the composite node C_{K-2} and sibling queue K and their associated synchronization queues.

In the above reduction procedure, we applied decomposition and aggregation starting from sibling queue 1 and proceeding sequentially to sibling queue K-1. However, it is not necessary that this order should be followed. For instance, we can partition the K-1 sibling queues into 2-sibling subnetworks and represent each of these by a separate flow-equivalent server. Pairs of these flow-equivalent servers may, in turn, be represented by a flow-equivalent server, and so on. We analyzed the K-sibling fork/join queue by pairing the sibling queues in different ways, and the results showed that the difference between them was trivial. This is of course hardly surprising, seeing that originally the sibling queues were numbered arbitrarily.

Step 2. In the original queueing network system substitute the K-sibling fork/join queue by the 2-sibling fork/join queue obtained from step 1 (see figure 3). Solve this model numerically to obtain the system throughput, the mean queue length of the fork node, the Kth sibling queue, and its synchronization queue. Also, by applying Little's relation one can obtain the mean waiting time in each of these three queues, and the mean response time of the fork/join operation, i.e., the mean time elapsing from the moment a job is split to the moment it joins again the fork node.

The above procedure yields performance measures for the fork node and the Kth sibling queue (and its synchronization queue). We can obtain performance measures of any other sibling queue i, i = 1,2,...,K-1, and its associated synchronization queue, by simply exchanging sibling queue i with sibling queue K and then apply the above algorithm.

The approximation procedure is based on the concept of "weakly coupled" systems. Hence, its accuracy depends only on the coupling between subsystems. 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. For presentation purposes, figures 5 to 19 give plots of the relative error ([(exact (or simulation) - approximate)/exact (or simulation)] 100%) for the system throughput, the mean queue lengths and the mean response time of the fork/join operation for the test

cases shown in table 1 (More detailed comparisons are presented in Liu [10]).

Case 1 :	3-sibling homogeneous fork/join model, $\mu_1 = \mu_2 = \mu_3 = 2$, M = 10. See figures 5, 6.
Case 2 :	3-sibling non-homogeneous fork/join model, $\mu_1 = 1$, $\mu_2 = 2$, $\mu_3 = 3$, M = 10. See figures 7, 8, 9.
Case 3 :	4-sibling homogeneous fork/join model, $\mu_1 = \mu_2 = \mu_3 = \mu_4 = 1$, M = 5. See figures 10, 11.
Case 4 :	4-sibling non-homogeneous fork/join model, $\mu_1 = 1$, $\mu_2 = 2$, $\mu_3 = 3$, $\mu_4 = 4$, M = 5. See figures 12, 13, 14.
Case 5 :	8-sibling homogeneous fork/join model, $\mu_1 = \mu_2 = ... = \mu_8 = 1$, M = 15. See figures 15, 16.
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 figures 17, 18, 19.

Table 1 : 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 homogeneous fork/join model with M = 10 jobs, the CPU time of the exact numerical procedure was 179.4 seconds, whereas the CPU time of the approximation procedure

was 8.73 seconds. For the 4-sibling homogeneous fork/join model with M = 5 jobs, the CPU time of the exact numerical procedure 112.3 second, and the CPU time of the approximation procedure was 1.982 second.

The approximation procedure gives good results for the system throughput and the mean response time of the fork/join operation for both homogeneous and non-homogeneous cases. We note that the relative error of the mean queue lengths increases as K increases and $\mu_0/\mu_1 \leq 1$.

We have observed empirically that the procedure gives a lower bound of the system throughput. This is due to the additional synchronization delay introduced each time a composite node is used to substitute a 2-sibling fork/join queue. Another contributing factor may be the fact that the state-dependent service rate of a composite node not only depends on the state of the composite node, but it also depends on the other siblings, a fact that was ignored in this approximation. Obviously, this solution gives an upper bound of the mean response time of the fork/join operation.

4. A Modified Approximation Procedure for Homogeneous Fork/Join Systems

In this section, we present a modified procedure for analyzing the homogeneous fork/join system shown in figure 1. The analysis is based on the empirical observation that the difference between the exact throughput $T_n(i)$ of the n-sibling fork/join queue (shown in figure 2) and the approximate throughput

$\hat{T}_n(i)$ obtained from step 1 of the above approximation algorithm, proportionally increases as n increases. In particular, we compared the exact throughput of $T_3(i)$ with the approximate throughput $\hat{T}_3(i)$ obtained from step 1, for i = 1, 2,...,M, and we found that the ratio of $T_3(i)/\hat{T}_3(i)$ is as follows,

$$T_3(i)/\hat{T}_3(i) \approx 1+\alpha(i).$$

The quantity $\alpha(i)$ can be seen as the error introduced by the approximation procedure. Likewise, the ratio of $T_4(i)$ and $\hat{T}_4(i)$ is given approximately by 1 + 2 $\alpha(i)$. In general, we have

$$T_n(i) \approx \hat{T}_n(i)[1+(n-2)\alpha(i)], \quad i = 1,2,....M.$$

where

$$\alpha(i) = T_3(i)/\hat{T}_3(i) - 1,$$

and n = 3, 4,...,K.

Therefore, we can numerically analyze the 3-sibling fork/join queue, in order to obtain exactly $T_3(i)$, i = 1, 2,...,M. We can also obtain $\hat{T}_3(i)$, i = 1, 2,..,M, approximately from step 1 of the above approximation algorithm. Using these two values, we can obtain $\alpha(i)$, for i = 1,2,...,M, which can then be used to construct an improved estimate of the system throughput, $T_{e,K}(i)$, i = 1,2,....M, of

the final 2-sibling fork/join queue obtained from step 1. We have

$$T_{e,K}(i) = \hat{T}_K(i)[1+(K-2)\,\alpha(i)], \quad i=1,2,\ldots,M. \tag{11}$$

Before, we proceed with step 2, we need to adjust the state-dependent service rate $\mu_c(i)$, i = 1, 2,...,M, of the composite queue in the final 2-sibling fork/join queue so that the system throughput is equal to $T_{e,K}(i)$, i = 1, 2,...,M. This is calculated as follows :

1. set m = 1;

2. set i = 0; $\mu_c^{(i)}(m) = \mu_c(m)$; $T_K^{(i)}(m) = \hat{T}_K(m)$.

3. $\mu_c^{(i+1)}(m) = \dfrac{T_{e,K}(m)}{T_K^{(i)}(m)} * \mu_c^{(i)}(m)$. Use $\mu_c^{(i+1)}(m)$ to obtain the adjustment throughput $T_K^{(i+1)}(m)$ from equation (10).

4. set i := i + 1; if $|\, T_{e,K}(m) - T_K^{(i)}(m)\,| < \epsilon$, then set $\mu_c(m) = \mu_c^{(i)}(m)$ and go to step 5, else go to step 3.

5. if $m < M$, then m := m + 1, go to step 2, else stop.

We can now apply the same numerical procedure as in step 2 to obtain the approximate solution.

This modified approximation algorithm was implemented on a VAX-11/785 in order to analyze the homogeneous fork/join model shown in figure 1. The main results obtained is the system throughput, and the mean queue length of the fork node, of a sibling queue and a synchronization queue. From this, other

performance measures, such as the mean response time of the fork/join operation and the mean synchronization time for each sibling, can also be obtained. The approximate results were compared against exact numerical values for 4-siblings, and with simulation results for 8-siblings. For presentation purposes, figures 20 to 23 give plots of the relative error ([(exact (or simulation) - approximate)/exact (or simulation)] 100%) for the system throughput, the mean queue length and the mean response time of the fork/join operation for the test cases shown in table 2 (More detailed comparisons are presented in Liu [10]).

Case 7 :	4-sibling homogeneous fork/join model, $\mu_1 = \mu_2 = \mu_3 = \mu_4 = 1$, M = 5. See figures 20, 21.
Case 8 :	8-sibling homogeneous fork/join model, $\mu_1 = \mu_2 = ... = \mu_8 = 1$, M = 15. See figures 22, 23.

Table 2 : Validation Test Cases

The modified procedure gives very good results for the system throughput and the mean response time of the fork/join operation (the relative error is less than 3%). But, we get a big error for the mean queue lengths of a sibling queue and its synchronization queue when $\mu_0/\mu_1 > 1$. We note, however, that the sum of these two mean queue lengths is practically equal to the exact solution.

5. Conclusion

We developed an approximation procedure for analyzing a closed queueing network with a K-sibling fork/join queue. The approximation procedure is based on decomposition and aggregation. 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. The modified procedure is only applicable to the homogeneous case. It gives very good results for the system throughput and 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 network consisting of the original fork node and a composite node with a state-dependent service rate equal to $T_{e,N}(i)$. This closed queueing network can be easily analyzed. The system throughput, the mean queue length of the fork node and the mean response time of the fork/join operation obtained from this model are very close to the exact results. The main drawback of this approach is that we loose information regarding the synchronization delay.

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.

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

[5] **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.

[6] **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.

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

[8] **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.

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

[10] **Liu, Y.C.**, "Performance Modeling of Distributed and Parallel Processing Systems", Ph.D. Dissertation, Department of Electrical and Computer Engineering, North Carolina State University, May 1988.

[11] **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.

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

[13] **Perros, H.G., Nilsson, A.A. and Liu, Y.C.**, "Approximate Analysis of Product Form Type Queueing Networks with Blocking and Deadlock", Performance Evaluation 8, 1988, 19-39.

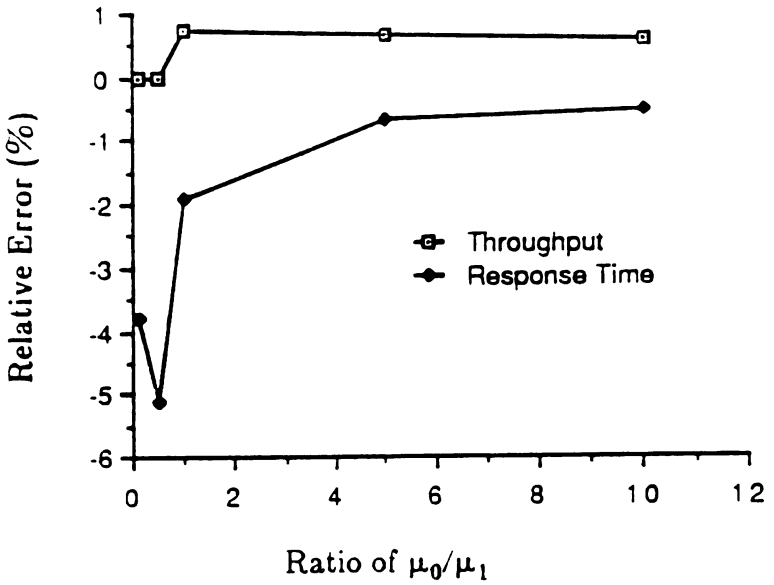

figure 5 : Relative Error of the Throughput and the Response Time for Case 1.

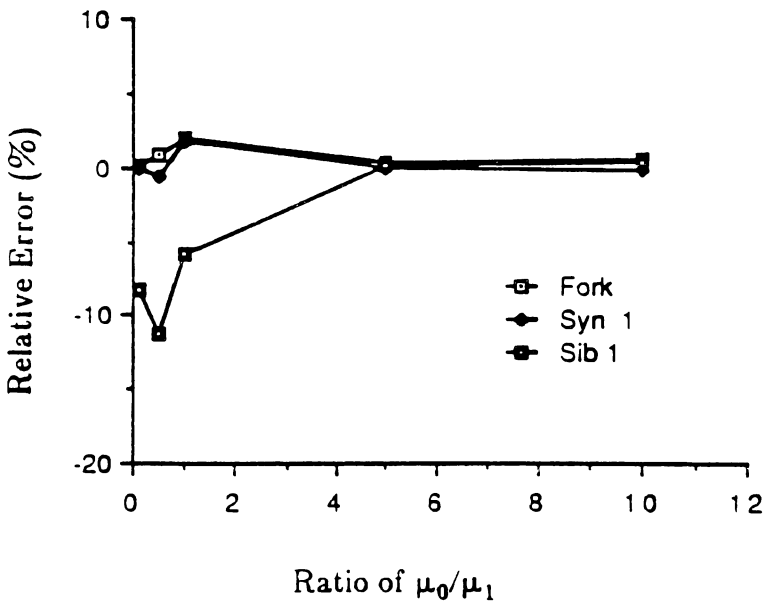

figure 6 : Relative Error of the Mean Queue Length for Case 1.

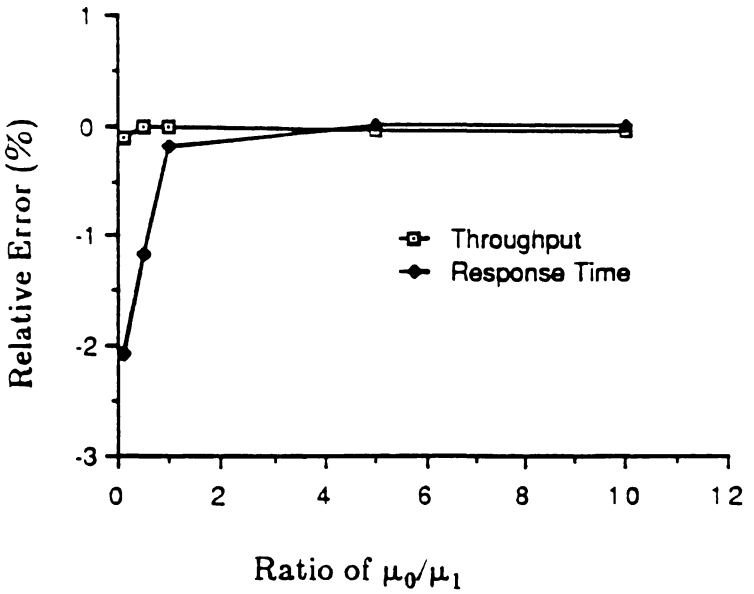

figure 7 : Relative Error of the Throughput and the Response Time for Case 2.

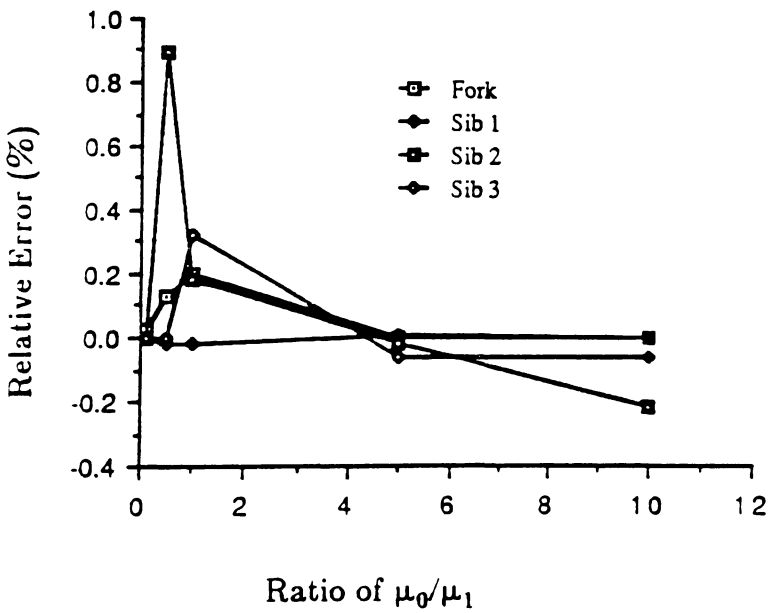

figure 8 : Relative Error of the Mean Queue Length of the Fork Node and Sibling Queues for Case 2.

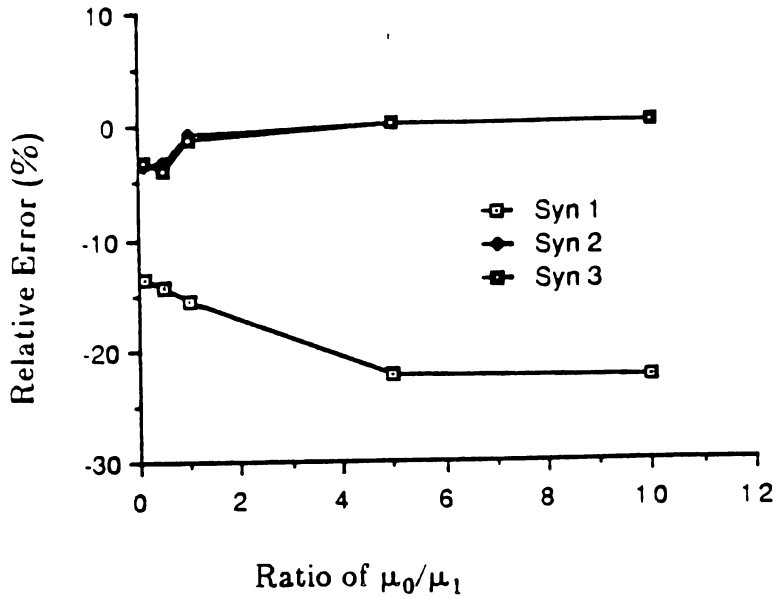

figure 9 : Relative Error of the Mean Queue Length of the Synchronization Queues for Case 2.

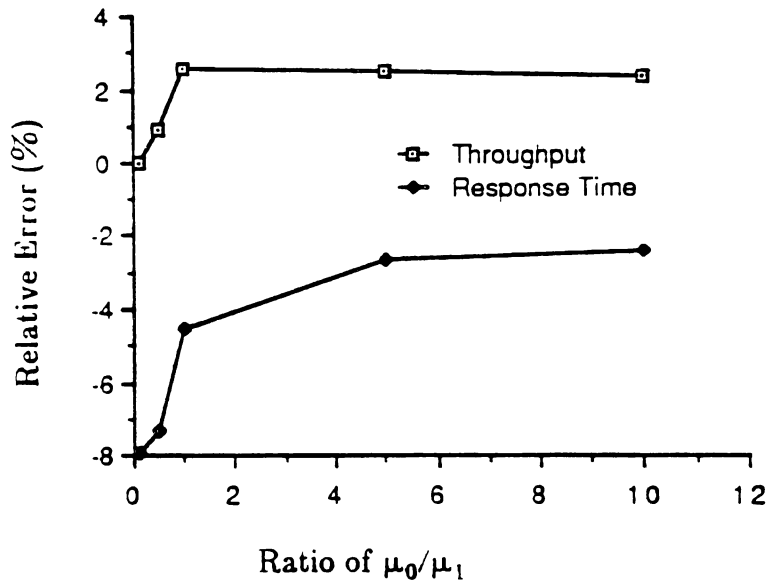

figure 10 : Relative Error of the Throughput and the Response Time for Case 3.

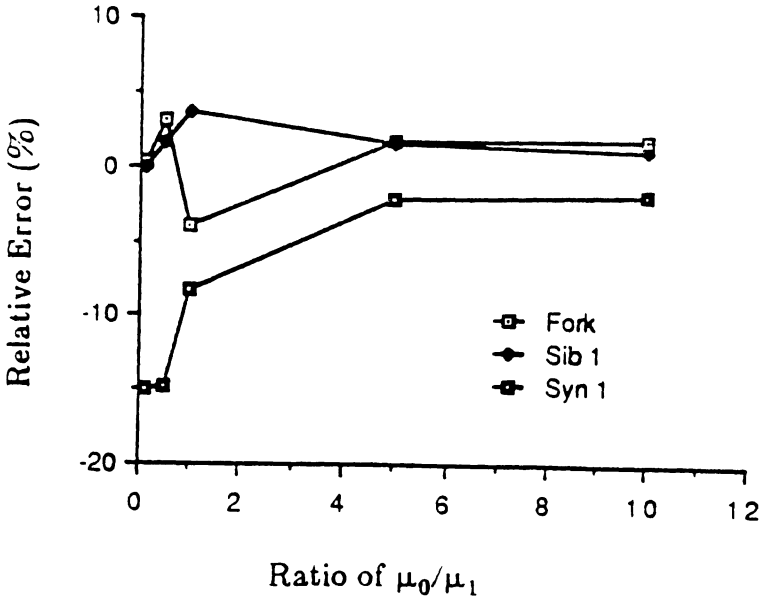

figure 11 : Relative Error of the Mean Queue Length for Case 3.

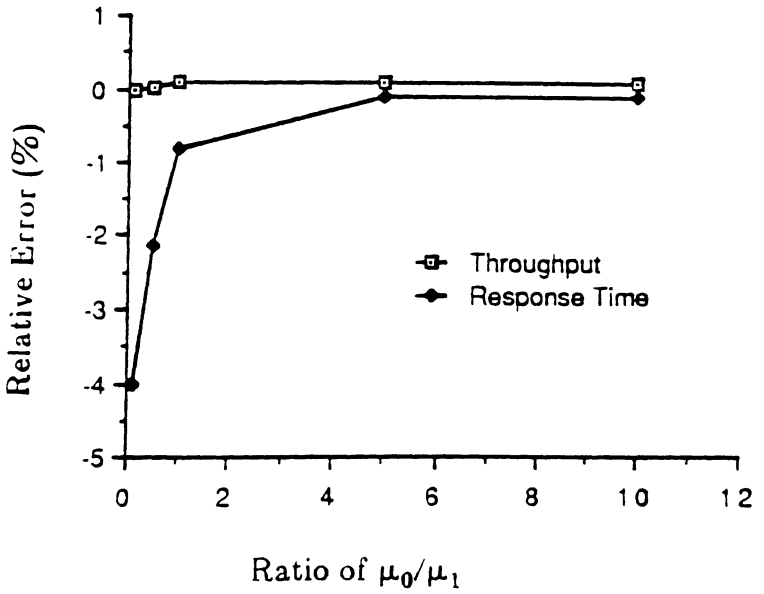

figure 12 : Relative Error of the Throughput and the Response Time for Case 4.

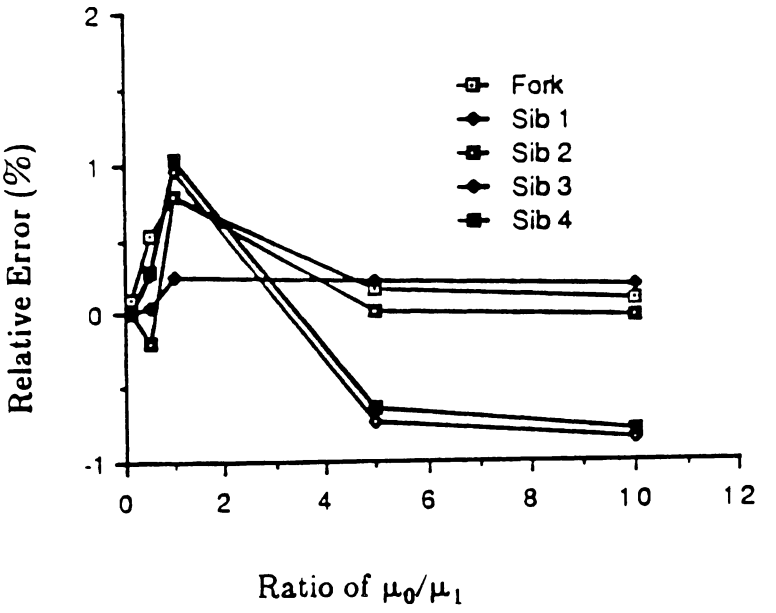

figure 13 : Relative Error of the Mean Queue Length of the Fork Node and Sibling Queues for Case 4.

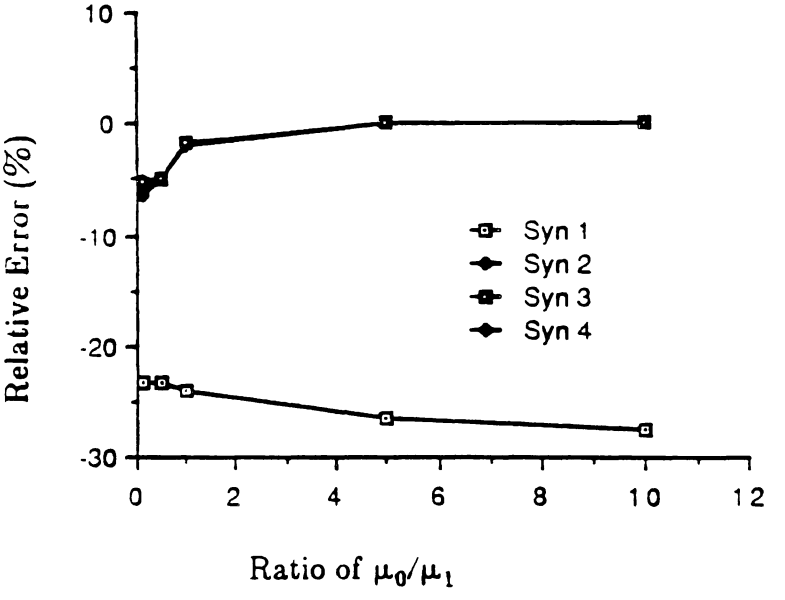

figure 14 : Relative Error of the Mean Queue Length of the Synchronization Queues for Case 4.

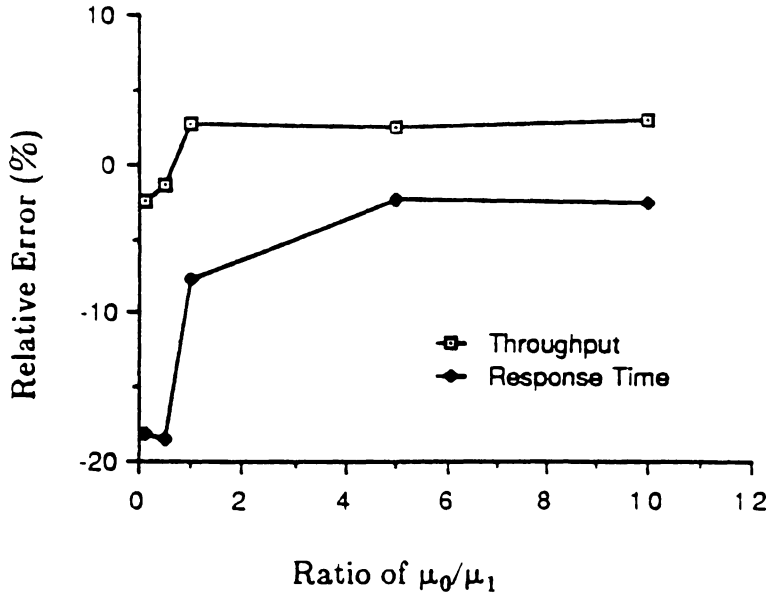

figure 15 : Relative Error of the Throughput and the Response Time for Case 5.

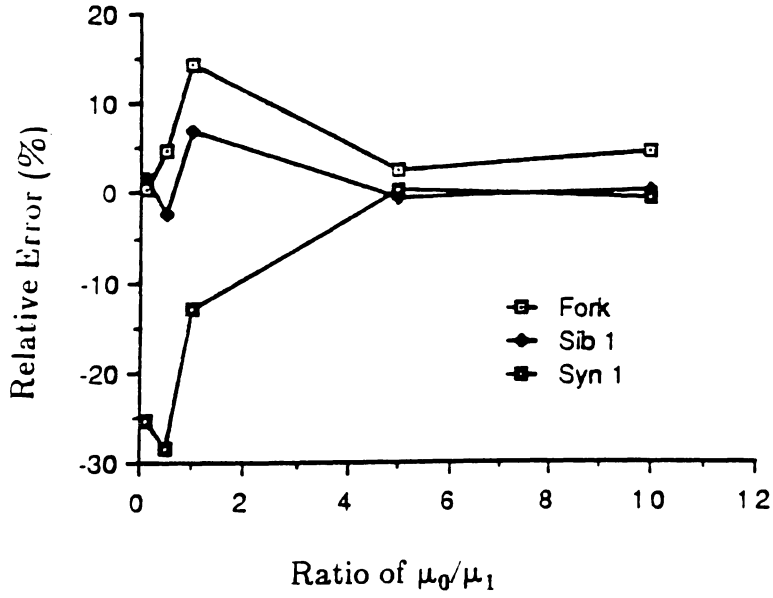

figure 16 : Relative Error of the Mean Queue Length for Case 5.

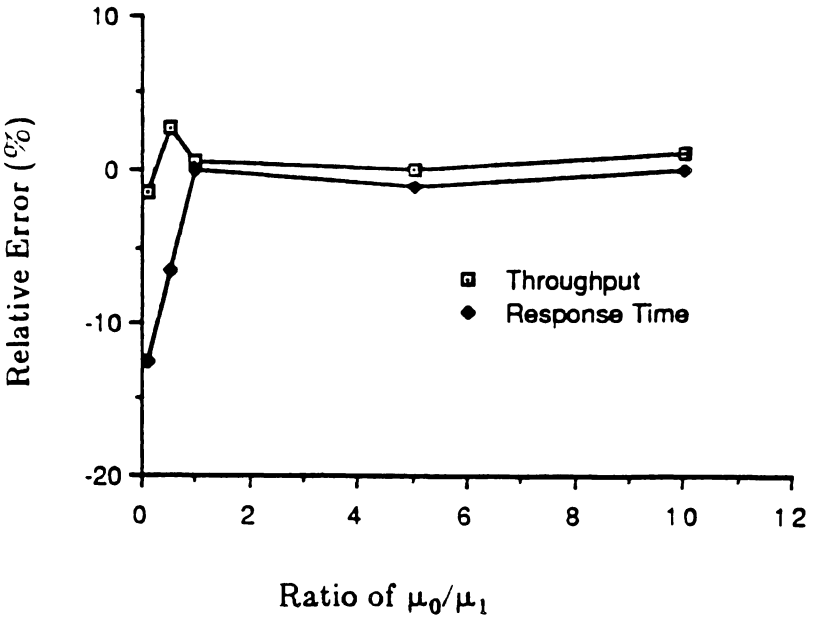

figure 17 : Relative Error of the Throughput and the Response Time for Case 6.

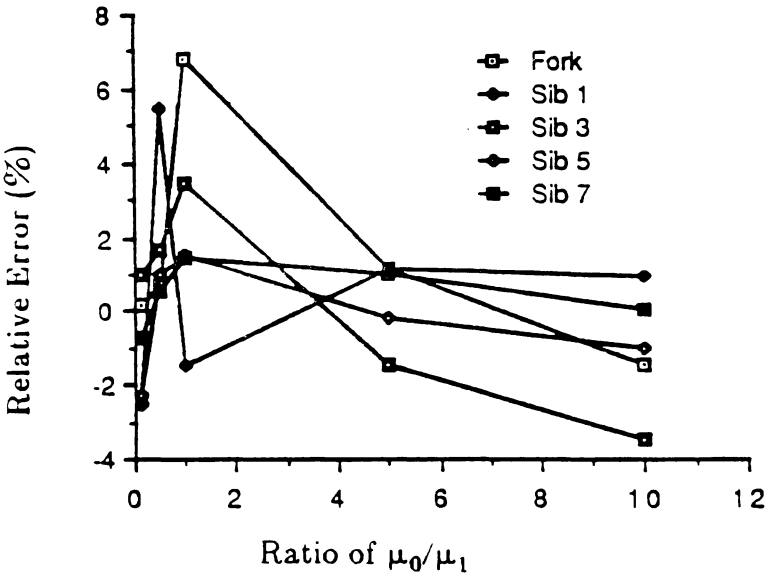

figure 18 : Relative Error of the Mean Queue Length of the Fork Node and Sibling Queues for Case 6.

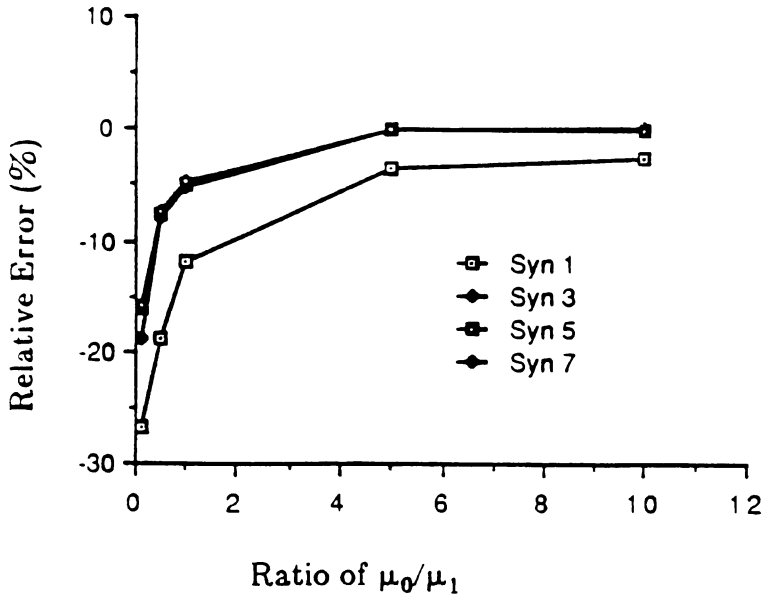

figure 19 : Relative Error of the Mean Queue Length of the Synchronization Queues for Case 6.

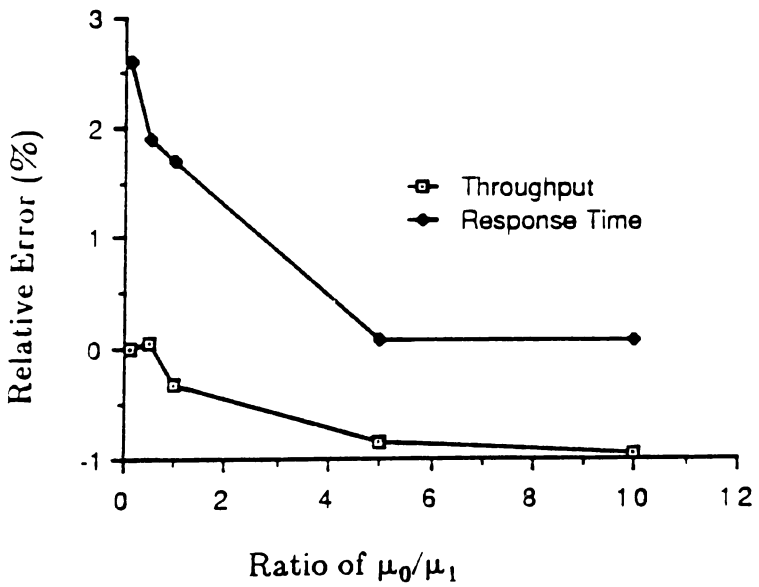

figure 20 : Relative Error of the Throughput and the Response Time for Case 7.

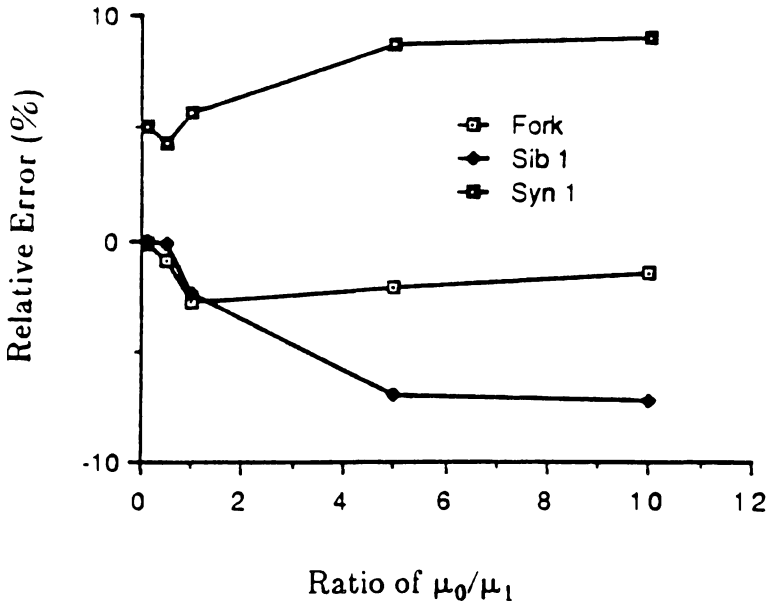

figure 21 : Relative Error of the Mean Queue Length for Case 7.

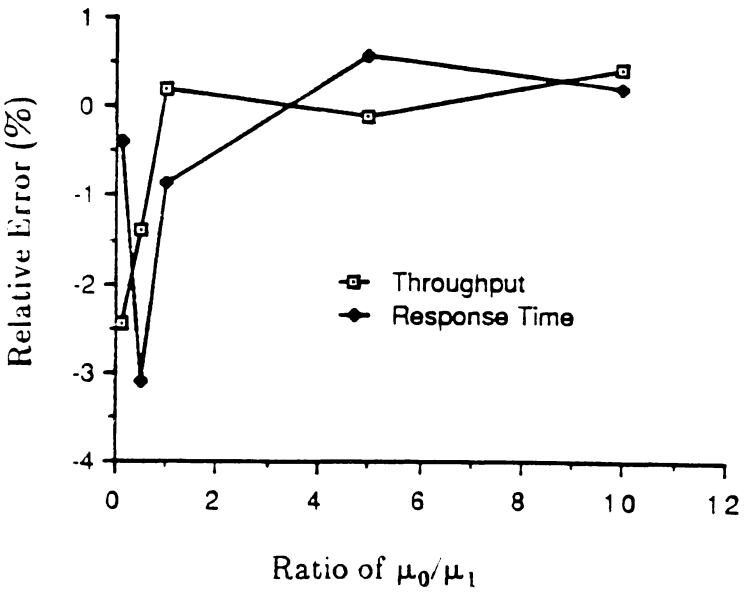

figure 22 : Relative Error of the Throughput and the Response Time for Case 8.

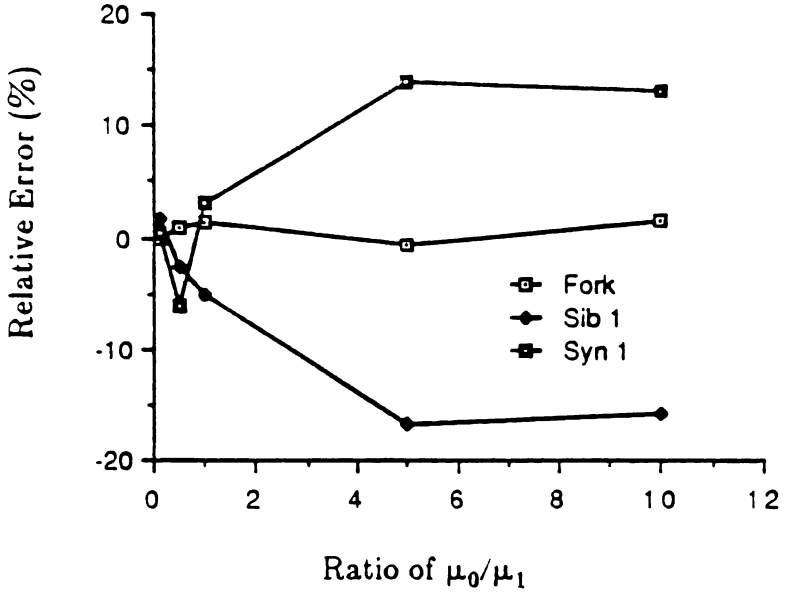

figure 23 : Relative Error of the Mean Queue Length for Case 8.