Analysis of Balanced Fork-Join Queueing Networks

Elizabeth Varki Lawrence W. Dowdy
Department of Computer Science
Vanderbilt University
Nashville, TN 37235
email: [varki,dowdy]@vuse.vanderbilt.edu

Abstract

This paper presents an analysis of closed, balanced, fork-join queueing networks with exponential service time distributions. The fork-join queue is mapped onto two non-parallel networks, namely, a serial-join model and a state-dependent model. Using these models, it is proven that the proportion of the number of jobs in the different subsystems of the fork-join queueing network remains constant, irrespective of the multiprogramming level. This property of balanced fork-join networks is used to compute quick, inexpensive bounds for arbitrary fork-join networks.

1 Introduction

Multiprocessor systems greatly influence the design of software processes. In uniprocessor systems, performance is limited by the speed of the single processor. This limitation is removed in multiprocessor systems by dividing a job into a number of tasks which are executed concurrently on the processing units. The parallel program has a simple fork-join structure. A job arriving at the system *forks* into various independent tasks. On completing execution, a task waits at the *join* point for its sibling tasks to complete execution. A job leaves the system once all its tasks complete execution. Although the job structure is simple, the synchronization constraints introduced by the forking and joining of tasks makes the modeling and analysis of multiprocessor systems complicated.

Performance analysis of parallel programs is important as is the development of tools for analyzing their performance. Exact solutions for the steady state distribution have been provided only for simple cases ([1], [3]). An exact response time analysis of a two server fork-join queue is given in [8]. Due to the difficulty of analyzing fork-join models exactly, many studies on fork-join queues concentrate on approximation techniques.

This paper introduces a quick and inexpensive bounding technique for obtaining performance measures of fork-join systems. The bounding technique is referred to as *balanced job bounds for fork-join* (BJB-FJ) systems and is analogous to balanced job bounds developed for product form networks [9]. The analysis given here is specific to 2-server fork-join systems but could be extended to more general fork-join networks. The main contributions of this paper are: (1) an exact analysis of the fork-join model using two non-parallel, equivalent models of the fork-join network, (2) a proof that the proportion of jobs in each of the subsystems of a balanced fork-join model remains constant, irrespective of the multiprogramming level, and (3) the introduction of a balanced job bounds technique for fork-join systems.

The remainder of the paper is organized as follows. Section 2 introduces the fork-join system under consideration, states the assumptions, and gives the notation. In Section 3, a detailed analysis of the fork-join model is given. Section 4 explains the balanced job bounds tech-

SIGMETRICS 96-5/96 Philadelphia, PA, USA

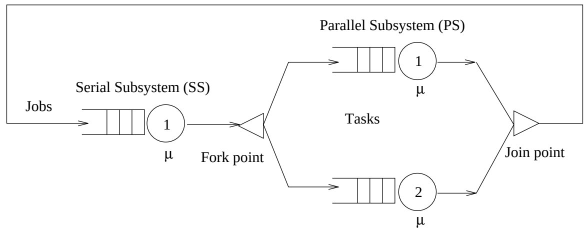

Figure 1: Balanced Fork-Join Queueing Network Model

nique for fork-join systems. Further research problems and extensions are discussed in Section 5.

2 The Balanced Fork-Join Queueing Network Model

The fork-join queueing network model studied in this paper is shown in Figure 1. A fixed number of identical *jobs* (referred to as the multiprogramming level) circulate in the network. The network consists of two interconnected *subsystems*, where each subsystem consists of one or more independent queueing systems. Each queueing system consists of a single service center and an infinite capacity queue. All servers have a first-come-first-served (FCFS) queueing discipline and all demands at the service centers have a negative exponential distribution with mean μ^{-1}. There are two types of subsystems: 1) a *serial* subsystem consisting of a single queueing system, and 2) a *parallel* subsystem consisting of two identical queueing systems. Upon arrival at a parallel subsystem, a job instantaneously forks into two independent and identical tasks, where task k, $k = 1, 2$, is assigned to the kth queueing system. On completing execution at the kth service center, the task waits at the *join* point for its sibling task to complete execution. A job leaves the subsystem as soon as all its tasks complete execution (i.e., arrive at the join point). A queueing network with serial and parallel subsystems is referred to as a *fork-join queueing network* (FJQN). A queueing network with only serial subsystems is referred to as a *serial network*. A *customer* in the FJQN refers to a job/task waiting for, or receiving, service at one of the service centers. Therefore, a customer at a service center in a serial subsystem refers to a job, while that at a service center in a parallel subsystem refers to a task.

Each service center in a parallel subsystem is visited by exactly one task of every job arriving at the subsystem. Thus, the visit ratio of a task to a service center in a parallel subsystem is equal to the visit ratio of the job to the subsystem. For the FJQN shown in Figure 1, the visit ratios of a job to each of the two subsystems are equal and, therefore, the loadings at all service centers are equal to μ^{-1}, the mean service time at a server. A network, where the loadings at all service center are equal, is called balanced.

Definition A *balanced* FJQN is one in which all jobs exhibit balanced resource usage (i.e., networks in which the loadings at all service centers are equal).

For the model shown in Figure 1, the serial subsystem is referred to as SS, and the parallel subsystem is referred to as PS. The notation used in this paper is:

- n_k: Number of tasks at the kth queueing station of a subsystem.
- $s, s' = (n_1, n_2)$: Vector representing the state of the parallel subsystem.
- $t, t' = (n_1)$: Vector representing the state of the serial subsystem.
- $S, S' = (t, s)$: Vector representing the state of the network.
- $J(s)$: Number of jobs in the parallel subsystem, given subsystem state s. $J((n_1, n_2))$ = maximum $\{n_1, n_2\}$.
- $j(s)$: Number of jobs having a task waiting at the join point, given subsystem state s. $j((n_1, n_2)) = |n_1 - n_2|$.
- $p_i(j)$: Probability that a job arriving at subsystem i sees j jobs in the subsystem just before arrival. The arriving job does not see itself in the subsystem.
- $RT_i(m)$: Mean response time at subsystem i when the multiprogramming level is m.
- $ST_i(m)$: Mean service time at subsystem i when the multiprogramming level is m.
- $Q_i(m)$: Mean queue length at subsystem i when the multiprogramming level is m.

- $U_i(m)$: Average utilization of subsystem i when the multiprogramming level is m.
- $TPUT(m)$: Mean system throughput when the multiprogramming level is m.
- H_k: The kth harmonic number defined as $\sum_{i=1}^{k} \frac{1}{i}$.

Note that since the system is analyzed at steady state, time is factored out of the notation. The set of states of the underlying queueing model of the FJQN forms an irreducible Markov process which is analyzed in the next section.

3 Analysis of Balanced FJQNs

In this section, a simple analysis of a fork-join network is presented. The fork-join network is mapped onto two serial networks, namely, a serial-join model and a state-dependent model. Each of the these models, taken separately, is difficult to solve. However, the three models are equivalent and this equivalence is used to prove some properties of the fork-join queue.

3.1 A Serial-Join Model of the FJQN

A job arriving at the parallel subsystem forks into two tasks, one of which spends its entire time at the service center it is assigned to; while its sibling task spends only a certain percentage of time at its service center and the rest of its time waiting at the join point. Since both service centers are identical, either one could serve the first task of the job completing execution (or equivalently, the last task of the job completing execution) with equal probability. Hence, the average job queue length of a service center in the parallel subsystem is equal to the average of the queue lengths of the service centers serving the first task and the last task of the job completing execution.

The parallel subsystem can be mapped onto two serial subsystems, P_a and P_j, as shown in Figure 2. The service center of P_a is equivalent to a service center in the parallel subsystem of the original fork-join model. The performance measures at P_a correspond to the average statistics of the two service centers of the original

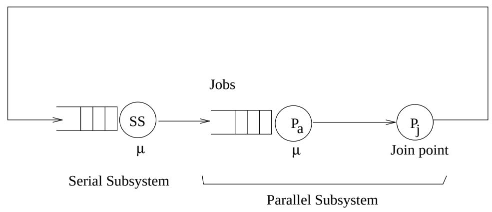

Figure 2: Equivalent Serial-Join Model of the FJQN

fork-join model. The service time at the second serial subsystem, P_j, is equal to the average delay encountered by a job at the join point of the original fork-join model. This equivalent model is referred to as the *serial-join* model. In the serial-join domain, a job arriving at the equivalent parallel subsystem does not fork into tasks. Instead, an arriving job spends some time in subsystem P_a, and then moves to the join point P_j. The queue length at P_a is equal to the average queue length at a service center of the original fork-join system, and this center services jobs at rate μ. The join point, P_j, is modeled as a delay server, and its service time varies according to the multiprogramming level. The join point, P_j, is analyzed in the following sections.

3.2 Markov Analysis

While the serial-join model allows one to view parallel subsystems without the forking of jobs into tasks, it doesn't simplify the analysis, since the service time at the join point is not known. Markov analysis is used here to identify and understand the properties of the fork-join network. First, consider the original fork-join queueing model when the multiprogramming level of the network is set to 1. The corresponding Markov process of the network has state space $\{(1,(0,0)),(0,(1,1)),(0,(0,1)),(0,(1,0))\}$. Figure 3(a) shows the state-space diagram of the Markov process. States $(0,(0,1))$ and $(0,(1,0))$ represent the network when there are no jobs at the serial subsystem, and the job at the parallel subsystem has one task at a service center while its sibling waits at the join point. These states have identical steady state probabilities and, hence, can be aggregated into a single state. The corresponding state-space diagram of the Markov process is given in Figure 3(b).

State S_1 represents the network when there is one

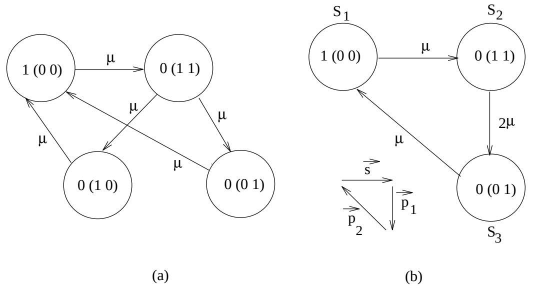

Figure 3: Markov Diagram of the Fork-Join Model at Multiprogramming Level 1

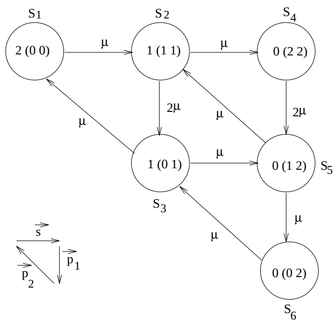

Figure 4: Markov Diagram of the Fork-Join Model at Multiprogramming Level 2

job in the serial subsystem. States S_2 and S_3 represent the network when there is one job at the parallel subsystem; S_2 represents the state when there are no tasks of the job waiting at the join point, and S_3 represents the state when one task of the job is at a service center while its sibling is at the join point. Transition arc $\vec{s}$ between states S_1 and S_2, represents the movement (at rate μ) of a job from the serial subsystem to the parallel subsystem. The transition arc $\vec{p_1}$ between states S_2 and S_3, represents the movement (at rate 2μ) of the first task of the job from a service center to the join point; while the transition arc $\vec{p_2}$ between states S_3 and S_1, represents the movement (at rate μ) of a job from the parallel subsystem to the serial subsystem.

Now consider the fork-join network when the multiprogramming level is set to 2. Figure 4 shows the state-space diagram of the underlying Markov process. Arcs $\vec{s}$ and $\vec{p_2}$ represent movement between the two subsystems, while $\vec{p_1}$ represents movement within the parallel subsystem. States S_2, S_3, S_4, S_5, and S_6 represent the network when there is at least one job in the parallel subsystem. Amongst these, states S_2 and S_4 represent the network when there are no jobs with a task waiting at the join point; states S_3 and S_5 represent the network when there is one job with a task at the join point; and state S_6 represents the network when there are two jobs which have a task waiting at the join point. The rates along transition arcs $\vec{s}$ and $\vec{p_2}$ are equal to μ, as in the case when the multiprogramming level was set to 1. These rates are state independent. However, the transition arc $\vec{p_1}$ between states S_5 and S_6 has rate μ, instead of 2μ. Thus, the movement of tasks to the join point slows down to rate μ whenever there are one or more jobs with tasks waiting at the join point.

The time spent by a job in the parallel subsystem can be factored into two phases. In *phase 1*, both tasks of the job are waiting for, or receiving, service at the service centers of the subsystem (i.e., there are 0 tasks of the job at the join point). In *phase 2*, only one task of the job is waiting for/receiving service at the service center while its sibling task waits at the join point.

3.3 State-Dependent Model of FJQNs

The properties of the parallel subsystem can be studied by viewing the subsystem from the perspective of the two phases of a job's response time. During the first phase, both tasks of the job are in the queues of the service centers. This phase ends with the movement of one of the tasks to the join point. In the remainder of the response time (i.e., phase 2), only one task of the job is at a service center, while its sibling waits at the join point. Movement out of phase 1 is depicted by downward transition arc $\vec{p_1}$, and movement out of phase 2 is depicted by upward transition arc $\vec{p_2}$. The time spent completing phase 1 and phase 2 of a job's response time in the parallel subsystem can be viewed as the time spent getting service at two service centers (or, serial subsystems), P_1 and P_2. Service time at server P_2 represents the time spent by a job in phase 2 of a job's response time. This server has a service time drawn from a negative exponential distribution with mean μ^{-1}. Service time at server P_1 represents the time spent by a job in phase 1 of a job's response time. The rate at which P_1 services jobs is dependent on the number of jobs in P_2. Service center P_1 services jobs at rate 2μ when there are no jobs in the queue of server P_2.

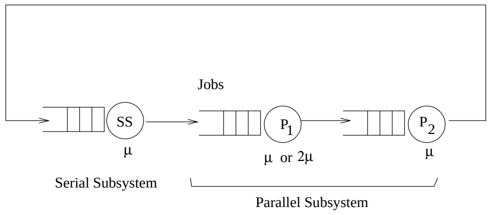

Figure 5: Equivalent State Dependent Model of the FJQN

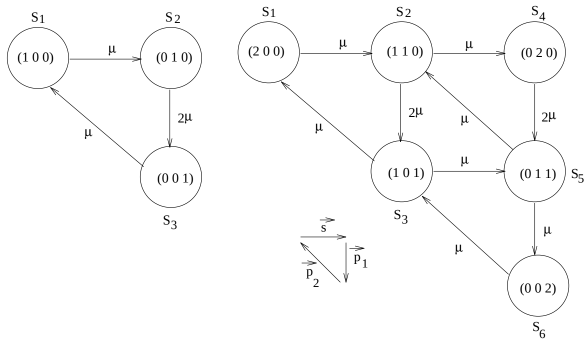

Figure 6: Markov Diagram of the Equivalent State Dependent Model at MPL of 1 and 2

However, the presence of one or more jobs at P_2 slows the service rate of P_1 to μ.

Viewing the parallel subsystem from the perspective of the two service centers P_1 and P_2 is similar to mapping the network to a different domain, one in which the fork-join model consists of three serial subsystems. The first serial subsystem in this domain is equivalent to the serial subsystem, SS, of the original fork-join model. The second and third serial subsystems are equivalent to subsystems with service centers P_1 and P_2, respectively, and they model the parallel subsystem. Since service time at service center P_1 is dependent on the number of jobs in service center P_2, the network is state-dependent. In this model, as in the case of the serial-join model, a job does not split into tasks in the parallel subsystem. The state-dependent model shown in Figure 5 is equivalent to the fork-join queueing network. The Markov diagram of the state-dependent model at multiprogramming levels of 1 and 2 is shown in Figure 6. A state, $(n'_1, (n_1, n_2))$, of the fork-join model is equivalent to the state $(n'_1, J(s)-j(s), j(s))$ of the state-dependent model.

3.4 Comparison of the Models

This section summarizes properties of the subsystems of the three equivalent models and shows the relationship between them. All equations stated here are direct consequences of the equivalence of the three models, and they link the models.

All of the models contain the serial subsystem, SS. This subsystem services jobs at rate μ, and its utilization $U_{SS}(m)$ equals $TPUT(m) * \mu^{-1}$. The other subsystems of the three models are:

- State-Dependent Model (Figure 5):

$$\begin{aligned} U_{P_2}(m) &= U_{SS}(m), \quad \forall m && (1)\\ Q_{P_2}(1) &= Q_{SS}(1) && (2)\\ U_{P_1}(1) &= 0.5 * U_{SS}(1) && (3)\\ U_{P_1}(m) &> 0.5 * U_{SS}(m), \quad \forall m > 1 && (4)\\ U_{P_1}(m) &< U_{SS}(m), \quad \forall m && (5) \end{aligned}$$

- Original Fork-Join Model (Figure 1): The equations given below relate the original fork-join model to the state-dependent model.

$$\begin{aligned} U_{SS}(m) &< U_{PS}(m), \quad \forall m && (6)\\ U_{PS}(1) &= U_{P_1}(1) + U_{P_2}(1) && (7)\\ U_{PS}(m) &< U_{P_1}(m) + U_{P_2}(m), \quad \forall m > 1 && (8)\\ Q_{PS}(m) &= Q_{P_1}(m) + Q_{P_2}(m), \quad \forall m && (9) \end{aligned}$$

- Serial-Join Model (Figure 2): The equations given below relate the serial-join model to the state-dependent model and the fork-join model. In particular, equations 12 and 13 link the three models of the fork-join network.

$$\begin{aligned} U_{P_a}(m) &= U_{SS}(m), \quad \forall m && (10)\\ Q_{P_a}(1) &= Q_{SS}(1) && (11)\\ Q_{P_a}(m) &= 0.5(Q_{PS}(m) + Q_{P_1}(m)), \forall m && (12)\\ U_{P_a}(m) &= 0.5(U_{PS}(m) + U_{P_1}(m)), \forall m && (13)\\ Q_{PS}(m) &= Q_{P_a}(m) + Q_{P_j}(m), \quad \forall m && (14)\\ Q_{P_j}(m) &= Q_{PS}(m) - Q_{P_a}(m), \quad \forall m &&\\ &= 0.5 * Q_{P_2}(m) && (15) \end{aligned}$$

3.5 Analysis of the Parallel Subsystem

In this section, the equivalent models are used to analyze the parallel subsystem of a FJQN. The service centers of the network are not necessarily balanced. However, all service centers of the parallel subsystem must be homogeneous.

Lemma 3.1 *The mean service time of subsystem P_1, $ST_{P_1}(m)$, is non-decreasing as the multiprogramming level, m, increases. In particular, $ST_{P_1}(1) < ST_{P_1}(m)$ for all $m > 1$.*

Proof: The service center in subsystem P_1 services jobs at rate 2μ when there are no jobs in subsystem P_2. The presence of one or more jobs in subsystem P_2 slows the service rate of P_1 to μ. Thus, the mean service time at P_1, $ST_{P_1}(m)$, is non-decreasing as the multiprogramming level increases. In particular, $ST_{P_1}(1) < ST_{P_1}(m)$ for all $m > 1$. □

Lemma 3.2 *The mean service time of parallel subsystem PS, $ST_{PS}(m)$, is non-increasing for increasing multiprogramming level m. In particular, $ST_{PS}(1) > ST_{PS}(m)$ for all $m > 1$.*

Proof: This result is proven in [8]. However, it can also be proved as follows. Equation 13 states that $U_{P_a}(m) = 0.5 * (U_{PS}(m) + U_{P_1}(m))$. This implies that $U_{PS}(m) = 2 * U_{P_a} - U_{P_1}(m)$. Lemma 3.1 shows that $ST_{P_1}(m)$ is non-decreasing for increasing multiprogramming level m. The mean service time of P_a, ST_{P_a}, is invariant of the multiprogramming level. The desired result follows directly. □

The following lemma is a direct consequence of lemmas 3.1, 3.2, and the equivalence of the fork-join and state-dependent model.

Lemma 3.3 *Consider the FJQN at any multiprogramming level $m > 1$. The increase in the service time of subsystem P_1 is equal to the decrease in the service time of parallel subsystem PS.*

Proposition 3.1 *If all service centers of a closed queueing network (not necessarily a FJQN) have service times drawn from a negative exponential distribution, then $U_i(m) < U_j(m) \Rightarrow Q_i(m) \leq Q_j(m)$; where i and j are subsystems/service centers of the network, and m is the multiprogramming level of the network.*

Proof: Assume that the visit ratios to subsystems i and j are equal. Then, $U_i(m) < U_j(m) \Rightarrow ST_i(m) < ST_j(m)$. It is also given that service times for both servers are drawn from the same distribution. The result follows. □

Note that if utilizations at two subsystems are equal, nothing can be said about their queue lengths in comparison to each other.

Lemma 3.4 *$Q_{P_2}(m) \leq 2 * Q_{P_1}(m)$, for all m.*

Proof: The lemma is proved by first showing that $Q_{P_2}(m) \leq Q_{P_a}(m)$, and then proving that $Q_{P_a}(m) \leq 2 * Q_{P_1}(m)$.

At multiprogramming level 1, $Q_{P_a}(1) = Q_{P_2}(1)$. For higher multiprogramming levels, Equation 8 states that $U_{PS}(m) < U_{P_1}(m) + U_{P_2}(m)$. This is equivalent to $U_{PS}(m) < U_{P_1}(m) + U_{P_a}(m)$. From Proposition 3.1, it follows that $Q_{PS}(m) \leq Q_{P_1}(m) + Q_{P_a}(m)$. But, by Equation 9, $Q_{PS}(m) = Q_{P_1}(m) + Q_{P_2}(m)$. Thus, $Q_{P_2}(m) \leq Q_{P_a}(m)$.

By Equation 12, $Q_{P_a}(m) = 0.5 * (Q_{PS}(m) + Q_{P_1}(m))$. It follows that $Q_{P_1}(m) = Q_{P_a}(m) - 0.5 * Q_{P_2}(m) \geq 0.5 * Q_{P_a}(m)$. The result follows. □

Theorem 3.1 *Consider the fork-join queueing network at multiprogramming level $m > 1$. The overall queue length at parallel subsystem, PS, is unaffected by the decrease in mean service time of PS (i.e., the increase in mean service time of P_1).*

Proof: First, consider the network at multiprogramming level 1. Since P_1 is twice as fast as P_2, therefore, $Q_{P_2}(1) = 2 * Q_{P_1}(1)$.

Now, consider the network at $m > 1$. Lemma 3.4 shows that $Q_{P_2}(m) \leq 2 * Q_{P_1}(m)$. Lemma 3.3 proves that the increase in mean service time of subsystem P_1 is proportional to the decrease in the mean service time of parallel subsystem PS. These two lemmas and Equation 9 ($Q_{PS}(m) = Q_{P_1}(m) + Q_{P_2}(m)$), imply that the percentage increase in the queue length of P_1 is proportional to the percentage decrease in the queue length of P_2, for $m > 1$ (i.e., the increase in the percentage queue length of P_1 is offset by a proportional decrease in the percentage queue length of P_2). Hence, the overall queue length at the parallel subsystem is unaffected by the decrease in mean service time of PS as the multiprogramming level increases beyond 1. $\square$

3.6 A Property of Balanced FJQNs

In balanced product-form networks, the utilizations at all service centers are equal, implying that the queue lengths at all service centers are equal, irrespective of the multiprogramming level. However, in balanced fork-join systems, the utilizations at parallel and serial subsystems are unequal. It follows that the queue lengths at serial and parallel subsystems are unequal. However, the utilizations at all the individual service centers of the FJQN are equal at all multiprogramming levels (i.e., there are no bottleneck service centers). The absence of bottleneck service centers in the balanced FJQN results in these networks satisfying the following interesting property: *In balanced fork-join systems, the proportion of number of jobs in each of the subsystems of the FJQN remains constant, and is invariant of the multiprogramming level.* This property of FJQNs is proved here using Theorem 3.1 and Lemma 3.5 given below.

Lemma 3.5 *There are no bottleneck service centers in the three equivalent models.*

Proof: This lemma holds trivially. The utilizations of the service centers in the serial and parallel subsystem of the FJQN are equal. Hence, there are no bottleneck service centers in the original fork-join model. The serial-join model and the state-dependent model are equivalent to the FJQN. It follows that there are no bottleneck servers in either of these models. $\square$

Note that a bottleneck service center is one in which the queue length grows at a faster rate than at any of the other centers. The presence of a bottleneck service center implies that as the multiprogramming level increases, the majority of customers will be found at this bottleneck center. The absence of bottleneck centers in product-form networks implies that the queue length at all service centers are equal. However, this need not be the case for non product-form systems.

Theorem 3.2 *For balanced fork-join queueing networks, $Q_i(m) = m * Q_i(1)$, for all multiprogramming levels m, and for all subsystems i.*

Proof: Consider the network when multiprogramming level is set to $m > 1$.

Theorem 3.1 shows that the increase in the percentage queue length of P_1 is offset by a proportional decrease in the queue length of P_2. Alternatively, the overall queue length at the parallel subsystem PS is not affected by the decrease in its mean service time as the multiprogramming level increases.

Also, Lemma 3.5 proves that there are no bottleneck service centers in the fork-join queueing network.

Theorem 3.1 and Lemma 3.5 imply that the mean queue length of serial subsystem, SS, and the parallel subsystem, PS, remains constant, regardless of the multiprogramming level m. $\square$

3.7 A Conjecture about the Arrival Instant Distribution

For closed FJQNs, it has been observed that the average number of jobs seen by a job arriving at a serial subsystem is less than the average queue length at the serial subsystem when the multiprogramming level of the network is one less.

Conjecture 3.1 *For a fork-join queueing network at multiprogramming level m, $Q_{SS}(m-1) > \sum_{k=0}^{m-1} k * p_{SS}(k)$, where $p_{SS}(k)$ is the probability that an arriving customer sees k jobs ahead of it in the serial subsystem. Equivalently, $Q_{PS}(m-1) < \sum_{k=0}^{m-1} k * p_{PS}(k)$, for the parallel subsystem PS.*

The observations shown in Table 1 refer to the balanced FJQN shown in Figure 1. The figures were obtained by solving the Markov diagram for the FJQN at various multiprogramming levels. The maximum difference between the arrival instant queue length and the queue length at multiprogramming level one less, is observed

m	$Q_{SS}(m-1)$	$\sum_{k=0}^{m-1} k * p_{SS}(k)$	Difference %
2	0.4	0.3846	4.00
3	0.8	0.7785	2.76
4	1.2	1.1769	1.96
5	1.6	1.5778	1.41
6	2.0	1.9969	0.16
7	2.4	≈ 2.4	≈ 0.0

Table 1: Observations backing the Conjecture in case of the balanced FJQN

at multiprogramming level 2. This difference decreases with increasing multiprogramming levels, and, in the limit, the difference between the two values approaches zero.

An intuitive argument for the validity of the conjecture is as follows. Suri, in [7], analyzes performance measures when the homogeneous service time assumption of product-form networks is violated. It is proven that an increase in the mean service time of a service center results in an increase in the proportion of customers at this center, and the steady state distribution of the network reflects this increase. Sevcik and Mitrani, in [6], show the relationship between an arriving customer's distribution and the steady state distribution of a closed arbitrary network. Thus, it seems to imply that an increase in the steady state probability would also result in an increase in the arrival instant distribution.

We have also observed that Conjecture 3.1 is valid for general K-sibling FJQNs ($K \geq 2$), not just for balanced, 2-sibling FJQNs. For networks with more than one serial and parallel subsystems, $Q_i(m-1) > \sum_{k=0}^{m-1} k * p_i(k)$, for all serial subsystems i. This implies that there exists at least one parallel subsystem j, such that $Q_j(m-1) < \sum_{k=0}^{m-1} k * p_{SS}(k)$. We are currently working on a formal proof of Conjecture 3.1.

3.8 Implications of the Mapping to Serial Systems

From the viewpoint of the underlying stochastic process of a queueing system, the main difference between serial networks and fork-join networks is the size of the step function of the state process. In serial networks, the number of customers in any of its subsystems can increase/decrease by at most one during an arrival/departure instant. In fork-join networks, the number of customers in a parallel subsystem increases by more than one at a job arrival instant. An important theorem in queueing theory states that: in any "system" (the actual nature of which is unimportant), and provided that the number of "customers" it contains varies by at most one at a time, the probability distribution of the number of customers in the system is the same just prior to an arrival and just after a departure. This theorem was proved by P.J. Burke in 1968 (unpublished), and can also be found in [2]. Burke's proof shows the generality of the result, which is not limited to queueing systems, but holds for any stochastic process in which realizations are step functions with only unit jumps. By mapping the fork-join network to serial networks, the theorem becomes valid for fork-join networks. (Alternatively, Burke's proof can be extended to directly show the equality of the arrival and departure instant distributions.)

4 Balanced Job Bounds for FJQNs

Balanced job bounds for product-form networks were developed by Zahorjan et. al. [9] as a technique for obtaining performance bounds efficiently, requiring few arithmetic operations. A brief explanation of the BJB solution technique is given here. A balanced product form network is one in which all jobs exhibit balanced resource usage. Because the service centers are all (effectively) identical, they have the same queue lengths, regardless of the multiprogramming level. If the multiprogramming level of the system is m, and if there are K service centers, then the queue length at each service center is equal to m/K. By applying the Arrival Theorem [5] and Little's Result, the throughput of a balanced network can be calculated as: $TPUT(m) = \frac{m}{D(K+m-1)}$, where D is the demand at each service center. This property of easily computing the performance of balanced networks is used for obtaining quick bounds for arbitrary product-form networks. The throughput of any given product-form network is bounded by the throughput of two systems: (1) lower bounds are obtained when the loadings at all service centers are raised to the maximum loading at any service center in the network, and (2) upper bounds are obtained when the loadings at all service centers are reduced to the minimum loading at any service center in the network. Tighter upper bounds are obtained using the average loading instead of the minimum loading.

The two properties of balanced product-form networks that are used to obtain performance bounds are: (1) the queue lengths at all service centers are equal at all multiprogramming levels, and (2) the average number of jobs seen by a job arriving at a service center is equal to the average queue length at the service center when the multiprogramming level of the network is one less [6], [4]. (This property is also referred to as the Arrival Theorem.) The corresponding properties of balanced fork-join queueing networks that can be used to obtain performance bounds are: (1) the percentage of the number of jobs (queue lengths) at each of the subsystems remains constant, irrespective of the multiprogramming level, and (2) the average number of jobs seen by a job arriving at a serial subsystem is less than the average queue length at the subsystem when the multiprogramming level is one less. The difference between the arrival instant and the steady state queue lengths decreases with increasing multiprogramming levels, and approaches zero in the limit (improving the accuracy of the bounds). These two properties of balanced fork-join queueing networks are used to compute performance measures of the balanced FJQN, at a given multiprogramming level m, in the following manner:

1. The mean queue length at each of the subsystems, at multiprogramming level 1, can be calculated exactly as follows:

   For serial subsystem SS, $RT_{SS}(1) = \frac{1}{\mu}$. For the parallel subsystem PS, response time is equal to the time taken to execute both tasks, which is equal to the second order statistic of the service time random variables. Thus, $RT_{PS}(1) = \frac{H_2}{\mu}$.

   The throughput and queue lengths can then be calculated as:

   $$\begin{aligned} TPUT(1) &= \frac{1}{RT_{SS}(1) + RT_{PS}(1)} \\ Q_{SS}(1) &= TPUT(1) \times RT_{SS}(1) \\ Q_{PS}(1) &= TPUT(1) \times RT_{PS}(1) \end{aligned}$$

2. The mean queue length at a multiprogramming level m is given by:

   $$Q_{SS}(m) = m \times Q_{SS}(1)$$
   $$Q_{PS}(m) = m \times Q_{PS}(1)$$

3. Close approximate values for throughput and response time at multiprogramming level m can be calculated using the exact queue length measurements computed at multiprogramming level $(m - 1)$. As shown in the previous section, it is observed that the Arrival theorem [5] is a close approximation for the response time of serial subsystems.

   $$\begin{aligned} RT_{SS}(m) &<\approx \frac{1}{\mu}(1 + Q_{SS}(m-1)) \\ TPUT(m) &= \frac{Q_{SS}(m)}{RT_{SS}(m)} \\ RT_{PS}(m) &= \frac{Q_{PS}(m)}{TPUT(m)} \end{aligned}$$

The performance measures of the balanced FJQN obtained using the Arrival theorem approximation are quite close to the actual performance values. Table 2 shows that the error between the approximate and the actual values is within 1.2%. This small error margin is not surprising, since the arrival instant queue length is very close to the queue length for the network at multiprogramming level one less (Refer to Table 1).

Performance measures for balanced fork-join networks can be easily computed at any given multiprogramming level using the method shown above. As in the case of product-form networks, performance bounds for an arbitrary fork-join queueing network, QN, can be obtained by constructing balanced FJQNs that bound the performance of QN, and solving these balanced networks [9]. In particular, the throughput (response time)

MPL	Actual Throughput	Approximate Throughput	%Error
2	0.5778	0.5714	1.10
3	0.6747	0.6667	1.19
4	0.7359	0.7273	1.16
5	0.7776	0.7692	1.08
6	0.8080	0.8000	0.99
7	0.8250	0.8235	0.18

Table 2: Actual and Approximate Throughput Values for the Balanced FJQN

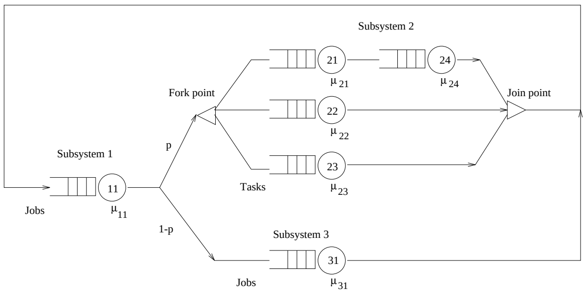

Figure 7: A General Fork-Join Queueing Network Model

of any given network, QN, is bounded by the throughput (response time) of two related systems, one in which the loadings at all service centers is raised to the maximum loading at any service center in QN, and another in which the loadings at all service centers is reduced to the minimum loading at any service center in QN.

5 Conclusions and Future Work

This paper gives an exact analysis of a closed, balanced FJQN. Two equivalent serial models of the FJQN are used in conjunction with the original fork-join model to analyze the parallel subsystem of the FJQN. It is shown that the probability distribution of the number of jobs in the system is the same just prior to an arrival and just after a departure. One contribution of this paper is the proof of the following property: the proportion of the number of jobs in each of the subsystems of a balanced FJQN remains constant, regardless of the multiprogramming level. In addition, it is observed that the average number of jobs seen by a job arriving at the serial subsystem is less than the average queue length at the subsystem when the multiprogramming level is decreased by one. These two properties of balanced FJQNs are used to compute quick bounds for arbitrary fork-join networks. The bounds are referred to as balanced job bounds for fork-join (BJB-FJ) networks since the bounding technique is similar to that of balanced job bounds for product-form networks.

While the analysis given here is specific to the model shown in Figure 1, it can be generalized to more general FJQNs of the type shown in Figure 7. A formal proof of the conjecture regarding the arrival instant distribution at subsystems of a FJQN is also required.

References

[1] Baccelli, F. "Two parallel queues created by arrivals with two demands: The M/G/2 symmetrical case", Report INRIA, 426, July 1985.

[2] Cooper, R. B. *Introduction to Queueing Theory*, Mercury Press/Fairchild Publications - a Capital Cities/ABC, Inc. company, MD, 3rd edition, 1990.

[3] Flatto, L., Hahn, S. "Two parallel queues created by arrivals with two demands I", SIAM J. Appl. Math., 44, Oct. 1984, pp. 1041 – 1053.

[4] Lavenberg, S.S., Reiser, M. "Stationary state probabilities at arrival instants for closed queueing networks with multiple types of customers", Journal of Appl. Prob., 17, Dec 1980, pp. 1048 – 1061.

[5] Reiser, M., Lavenberg, S.S. "Mean-value analysis of closed multichain queueing networks", Journal of the ACM, 27, 2, April 1980, pp. 313 – 322.

[6] Sevcik, K. C., Mitrani, I. "The distribution of queueing network states at input and output instants", JACM, 28, 2, April 1981, pp. 358 – 371.

[7] Suri, R. "Robustness of queueing network formulas", JACM, 30, 3, July 1983, pp. 564 – 594.

[8] Varki, E., Dowdy, L.W. "Response time analysis of two server fork-join systems", MASCOTS, 1996.

[9] Zahorjan, J., Sevcik, K.C., Eager, D.L., Galler, B. "Balanced job bound analysis of queueing networks", Comm. of ACM, 25, 2, 1982, pp. 134 –141.