Response Time Analysis of Two Server Fork-Join Systems

Elizabeth Varki and Lawrence W. Dowdy
Department of Computer Science
Box 1679–B, Vanderbilt University
Nashville, TN 37235
email: [varki,dowdy]@vuse.vanderbilt.edu

Abstract

This paper gives an exact analysis of mean response time of fork-join systems. The response time is shown to be the sum of parallel wait and parallel service times which are analogous to wait and service times in non-parallel systems. It is shown that the parallel service time of a job is dependent on the state at arrival time of the job. A formal parallel wait time analysis, similar to the wait time analysis of non-parallel systems is also given. The analysis is carried out using event/time sequence (ET-S) trees, a simple modeling technique introduced in this paper.

1 Introduction

Multiprocessor systems allow the design of parallel programs that can be split into tasks which are then executed concurrently on separate processing units. Upon completion, a task waits for its siblings to finish. A job completes execution only when all its tasks complete execution. This *fork-join* structure is a simple model of the behavior of parallel programs.

Due to the increasing use of multiprocessor systems, performance analysis of parallel programs is important. In particular, the execution time of a parallel job is an important performance measure and has been the topic of several previous studies. However, the synchronization introduced by forks and joins makes analysis very difficult. Exact solutions for the steady state distribution have only been provided for a two server case in [1], [2]. In [2], Flatto and Hahn consider a fork-join system with two heterogeneous servers. In [1], Baccelli considers the symmetric M/G/2 case. Due to the difficulty of analyzing fork-join models exactly, most studies on fork-join queues concentrate on approximation techniques.

This paper analyzes the response time for a job arriving at a two server fork-join queueing system by tracing the sequence of all possible events that can occur from the job's arrival until the job's departure. The network can be either open or closed. The focus of the paper is not the steady state/arrival instant distribution of the fork-join system. Interested readers are referred to [1] and [2] for such analysis. Instead, an exact analysis of the mean parallel service time and mean parallel wait time of a "tagged" job arriving at the fork-join system is given. The main contributions of this paper are: (1) formalizing the concepts of parallel wait and service times (i.e., pwait and pservice times, respectively) of a job in a fork-join queueing system, (2) showing that the parallel service time is dependent on the state at arrival instant of a job, (3) showing that if the parallel service time of a job includes only the execution time of the last task, then the execution time of the first task effectively vanishes from the response time analysis, (4) presenting an exact parallel wait time analysis of the fork-join system, and (5) introducing a simple technique, namely, event/time sequence (ET-S) trees, to analyze the overall mean response time of a job in a parallel system.

2 The Queueing Model

The fork-join system analyzed in this paper consists of two identical parallel queueing systems, referred to as a *parallel subsystem*. Each of the queueing systems consist of a single server and an infinite capacity queue. All servers have a first-come-first-served (FCFS) queueing discipline and all demands at the service center have a negative exponential distribution with mean μ^{-1}. The model can be either open or closed as shown in Figure 1. In the open system, the arrival of identical *jobs* to the parallel subsystem forms a Poisson process with mean λ^{-1}. In the closed system, a fixed number of identical jobs circulate in the network, and jobs arrive at the parallel subsystem from a central server which services jobs at rate λ. Upon arrival at the 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, a task waits at the

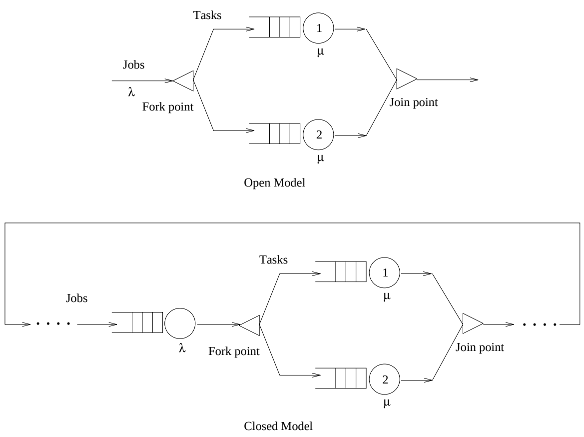

Figure 1: Queueing Models of the Fork-Join System

join point for its sibling task to complete execution. A job leaves the subsystem as soon as all its tasks complete execution (i.e., tasks corresponding to a job are joined before departing the subsystem).

The set of states of the underlying queueing model forms an irreducible Markov process. In the open system, it is assumed that $\lambda < \mu$, since the limiting distribution of the Markov process exists only if this condition holds. This paper analyses the mean response time of a *tagged* job that arrives at the parallel subsystem by tracing the sequence of all possible events that can occur until the job exits the system. Thus, the probability of the subsystem states seen at arrival instant of the tagged job is necessary to compute the overall mean response time of the parallel subsystem. In the open system, the distribution seen by an arriving customer is equal to the steady state distribution. Flatto and Hahn [2] obtain asymptotic formulas for the steady state probabilities of an open fork-join system with two heterogeneous servers. For the closed system, Sevcik and Mitrani [3] show the relationship between an arriving customer's distribution and the steady state distribution. However, as explained in Section 1, the focus of this paper is not the steady state/arrival instant distribution of the fork-join system, but rather the analysis of mean response time of a tagged job given the state at arrival instant of this job.

3 States of the Parallel Subsystem

The state of the parallel subsystem at time t, $S(t)$, is represented by a vector $(n_1(t), n_2(t))$ where $n_k(t)$ represents the number of tasks at service center k, $k = 1, 2$. Since the system is analyzed at steady state, time is factored out of the notation and $S(t)$ is referred to as S. The number of jobs, $J(S)$, in the parallel subsystem given $S = (n_1, n_2)$ is the maximum of $\{n_1, n_2\}$. The number of jobs which have a task waiting at the join point, $j(S)$, is equal to $|n_1 - n_2|$.

Since all servers in the parallel subsystem are identical, the limiting distribution for all states S', where S' is a permutation of the elements of the set $\{n_1, n_2\}$ are identical. Hence, all states of type S' can be aggregated into one state, thereby reducing the state space of the underlying Markov process. The limiting probability of this aggregate state is the sum of the limiting probabilities of all states S'. For example, states (0,1) and (1,0) represent states where one task of a job is executing while its sibling is waiting at the join point. These states have identical steady state probabilities and can be aggregated into a single state.

The notation used in this paper is given below.

- n_k: Number of tasks at the kth queueing station of the parallel subsystem.
- $S, T = (n_1, n_2)$: Vector representing the state of the parallel subsystem.
- $J(S)$: Number of jobs in the parallel subsystem, given subsystem state S.
- $j(S)$: Number of jobs having a task waiting at the join point, given subsystem state S.
- $S^{a+}, (n_1, n_2)^{a+}$: State of the parallel subsystem seen by a job just after arrival at the subsystem. The arriving job also sees itself in the subsystem. Thus, in state $(n_1, n_2)^{a+}$, $n_1 \geq 1$ and $n_2 \geq 1$.
- $S^{+}, (n_1, n_2)^{+}$: State of the parallel subsystem at an arbitrary instant. If the arbitrary instant is an arrival instant, then $S^{+} = S^{a+}$.
- $p(S^{a+})$: Probability that an arriving job to the parallel subsystem sees state S^{a+} just after arrival. For instance, $p((1,1)^{a+})$ is the probability that an arriving job encounters an empty system.
- $\rho_k(S^{a+})$: Probability that the trigger task (defined as the task that starts executing last from among its siblings) of an arriving job that sees the subsystem in state S^{a+} will execute in parallel with $(k-1)$ of its siblings.
- RT: Response time.
- WT: Wait time.
- ST: Service time.
- H_k: The kth harmonic number defined as $\sum_{i=1}^{k} \frac{1}{i}$
- $E[X]$: Expected value of a random variable X.
- $P(e)$: Probability of event e occurring.

4 Response Time Analysis – Terms and Issues

A job, upon arrival at the parallel subsystem, forks into two identical tasks, each serviced by a separate queueing system. Response time includes the time from the instant of job arrival until all tasks complete execution. For the duration of this paper, response time of an arbitrary tagged job is analyzed.

The **trigger** task of a tagged job refers to the task that *starts executing last* from among its siblings (i.e., it is the last sibling task of a job to reach the head of its service queue). The **first (second)** task of a job refers to the task that *completes* execution first (second) from among the sibling tasks of the job. The concepts pwait time and pservice time are formally defined here and are key in the analysis that follows.

Definition: *The* **parallel wait** *time of a job,* **pwait** *time, is the time taken from the instant a job arrives at the parallel subsystem until its trigger task starts executing. Thus, the pwait time of a job includes the time taken from the instant of job arrival until there are no tasks of the job waiting for service.*

Definition: *The* **parallel service** *time of a job,* **pservice** *time, is the time taken from the instant the trigger task starts executing, until the job leaves the join point (i.e., until the second task completes service).*

The response time of a job at the parallel subsystem is the sum of the pwait time and the pservice time. This is similar to the response time analysis of non-parallel (i.e., serial) systems. Wait time for a job arriving to a serial subsystem is the time taken to service the jobs ahead of it in the queue. This definition is analogous to the definition of pwait time. However, for serial subsystems, wait time of a job is also the time from arrival instant until the job starts executing. This is not the case with parallel subsystems, where pwait time is the time from arrival instant until the trigger task of the job starts executing. Due to the presence of two independent queueing stations, it is possible that when the trigger task of a job gets to the head of its service queue, its sibling has completed execution and is already at the join point. In this case, the execution time of this sibling (i.e., the first task) is included in the pwait time of the tagged job; and the pservice time of the job is the time taken to execute only the trigger task. The pservice time of a job is equal to the sum of the time taken to execute both tasks concurrently weighted by probability $\rho_2(S^{a+})$ and the time taken to execute just one task (i.e., the trigger task which is also the second task in this case) weighted by probability $\rho_1(S^{a+})$. It also follows that with probability $\rho_1(S^{a+})$, the execution time of the first task completely overlaps the pwait time of the job. In this case, the execution time of the job's first task completely overlaps the pservice time of some of the jobs which arrived ahead of the tagged job.

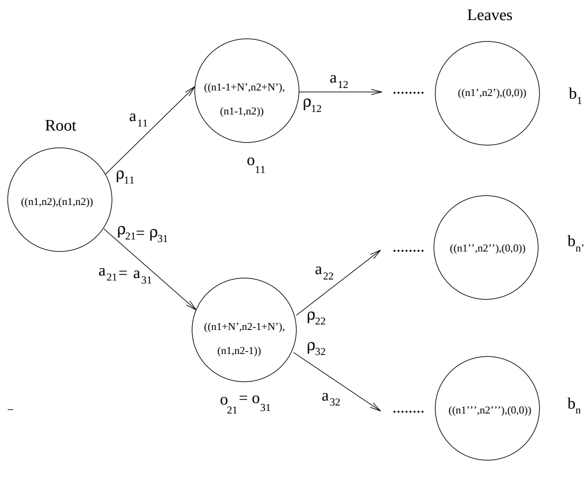

Figure 2: ET-S Tree

Two factors must be taken into account when calculating the pservice time of a tagged job in the parallel subsystem: (a) the probability of the trigger task executing either concurrently or non-concurrently with its sibling and, (b) the pservice time of the job once the trigger task reaches the head of its queue. The above informal analysis is made more precise in Section 6, using ET-S trees which are discussed in Section 5.

5 Event/Time Sequence Trees

An *Event/Time Sequence* (ET-S) tree is a formal, abstract model (i.e., a data structure representation) of the time spent by a job in a parallel subsystem given the state seen by a tagged job at arrival instant. It is developed as a method for analyzing the mean response time of a job in a parallel subsystem. The ET-S tree enumerates all of the possible sequences of events that can occur from the arrival time of a job to the parallel subsystem, until the instant that this tagged job leaves the subsystem. The mean time taken for each sequence of events is computed and weighted by the sequence probabilities.

Figure 2 is an ET-S tree diagram for the parallel subsystem. An ET-S tree is composed of a set of *branches*, $B = \{b_1, b_2, ..., b_n\}$, which are paths from the root to the n leaves of the tree. Each branch represents a separate event sequence, and all branches are assigned an arbitrary number. A branch, b_i, is a sequence of arcs and nodes and is represented by $((a_{i1}, o_{i1}), (a_{i2}, o_{i2}), ..., (a_{im}, o_{im}))$, where a_{ij} represents an arc and o_{ij} represents a node. Arc a_{i1} connects the root of the ET-S tree to node o_{i1}. Node o_{im} represents a leaf of the tree. The nodes in the tree represent the subsystem state with respect to a tagged job, x, whose response time is to be calculated. Each node, o_{ij}, in the tree is represented by a vector (S, R), where $S = (n_1, n_2)$ is the subsystem state and represents the number of tasks in each queueing station of the subsystem; and $R = (r_1, r_2)$ represents the position of tasks of job x in the two subsystem server queues. An ET-S tree is identified by its root, where the root (usually) represents the subsystem state seen just after arrival of job x to the subsystem. In Figure 2, the root is represented by $(S, R) = ((n_1, n_2), (n_1, n_2))$. The leaves of an ET-S tree represent subsystem states seen just after the departure of job x from the subsystem. Thus, the leaves have $R = (0, 0)$. A directed arc from node $o_{i(j-1)}$ to node o_{ij} represents the completion of service for a task at the head of one of the service queues.

If node $o_{i(j-1)}$ has a label of $((n_1, n_2), (r_1, r_2))$, then node o_{ij} will represent either $((n_1 - 1 + N', n_2 + N'), (r_1 - 1, r_2))$ or $((n_1 + N', n_2 - 1 + N'), (r_1, r_2 - 1))$, where N' is equal to the number of jobs that arrive at the subsystem during the service time of the task that just completed its service. The probability of either of these events occuring is equal to 0.5 since both servers are identical. The time taken for this transition is equal to the mean of the 1st-order statistic of the two exponential service time distributions. From a node having a label of $((n_1, 0), (r_1, 0)), n_1 \neq 0$, a transition occurs to the node $((n_1 - 1 + N', N'), (r_1 - 1, 0))$ with probability 1. The time taken for this transition is μ^{-1}. Thus, each arc, a_{ij}, of the ET-S tree has a probability, ρ_{ij}, associated with it. Also, each arc explicitly represents the mean time, t_{ij}, taken to transit from one node to another.

The mean response time of a job that sees the subsystem in state S^a just before arrival can be computed by using an ET-S tree with the root set to S^{a+}, where S^{a+} is the state seen immediately after the job arrives at the subsystem. The mean response time is equal to the weighted sum of the time required by each event sequence. That is, $E[RT|S^{a+}] = \sum_{b_i} \left[\left(\prod_j \rho_{ij} \right) \left(\sum_k t_{ik} \right) \right]$, where the range of i is over all event sequences and the ranges of j and k are over the length of sequence i. An example of an ET-S tree used in computing response time is given in Figure 3. The mean response time of a job that sees the subsystem in state (1,0) just before arrival is computed to be $\frac{9}{4\mu}$.

The height and width of an ET-S tree are dependent on the number of jobs waiting at the subsystem upon arrival, the number of service centers in the parallel subsystem, and the number of subsequent job arrivals. As the height and width increase, it becomes increasingly difficult to manage and analyze the ET-S tree. This problem can be solved by defining ET-S trees recursively. An *atom* is defined to be a fully enumerated ET-S tree with its associated mean response time. If a branch of another tree is identical to an atom, then the branch can be terminated at that point and the response time measures of the atom can be augmented to the branch. A leaf which is actually an atom is referred to as a *macro leaf* (see Figure 4). Any tree whose response time has been computed can be used as an atom.

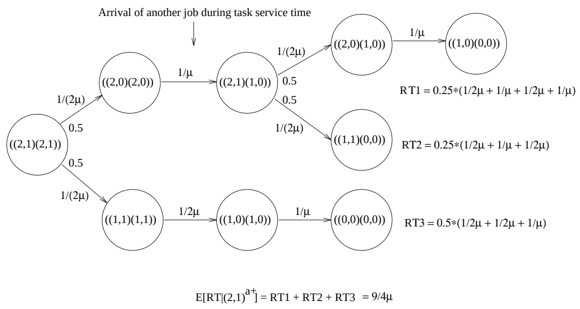

Figure 3: ET-S Tree Example

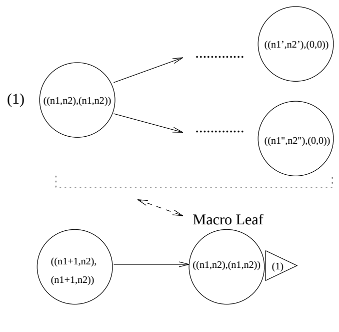

Figure 4: Recursive ET-S Trees

6 Pservice Time Analysis

The two issues raised in Section 4, namely, the effect on service time of a tagged job by previous and subsequent job arrivals are formalized in this section. For detailed proofs, interested readers are referred to [4].

Theorem 6.1 *The mean pservice time of a tagged job in the parallel subsystem is not affected by jobs that arrive subsequently.*

Proof: When the trigger task of a tagged job starts executing, the state is either $((N + 1, N + 1)(1, 1))$ where N is the number of jobs arriving subsequently

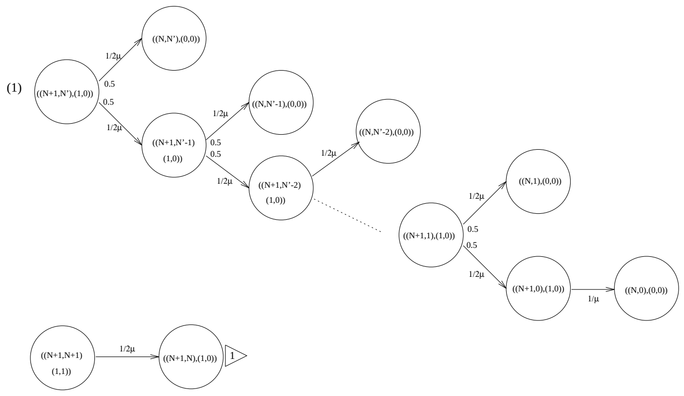

Figure 5: ET-S tree of the mean pservice time of a job

to the tagged job, or the system state is $((N+1, N')(1,0))$, where $N' \leq N$ and represents the state where some of the first tasks of subsequent arrivals have completed service. The ET-S trees of these two states are shown in Figure 5. The mean pservice time, $E[ST|N$ subsequent arrivals] calculated from the atom (1) in Figure 5 is given by,
$E[ST|N] = \left[\sum_{i=1}^{N'} \left(\frac{1}{2}\right)^i \frac{i}{2\mu}\right] + \left(\frac{1}{2}\right)^{N'} \left[\frac{N'}{2\mu} + \frac{1}{\mu}\right] = \frac{H_1}{\mu}$.
Similarly, $E[ST|N$ subsequent arrivals] calculated from the recursive ET-S tree in Figure 5 is $\frac{1}{2\mu} + \frac{1}{\mu} = \frac{3}{2\mu} = \frac{H_2}{\mu}$. Since $E[ST|N$ subsequent arrivals] is independent of N in all cases, the theorem holds. □

The following corollaries follow from Theorem 6.1.

Corollary 6.1 *If the trigger task of a tagged job executes alone (i.e., if the pservice time of a job includes only the execution time of a single task), then the execution time of the first task effectively vanishes from the mean response time analysis.*

Corollary 6.2 $\frac{1}{\mu} \leq E[ST] \leq \frac{3}{2\mu}$.

Corollary 6.3 $E[ST|S^{a+}] < E[ST|T^{a+}]$ *if and only if* $\rho_1(S^{a+}) > \rho_1(T^{a+})$.

Jobs that arrive after a tagged job, x, can be ignored in the response time analysis of job x since they do not affect the pservice time of any of the preceding jobs. Thus, for every node, $o_{ij} = (S, R)$, of an ET-S tree, the vector S is effectively reduced to R, and a node label can be represented by just the vector R.

The following theorem addresses the effect that the state seen at arrival instant (i.e., the jobs waiting ahead of the tagged job) has on the pservice time of a job.

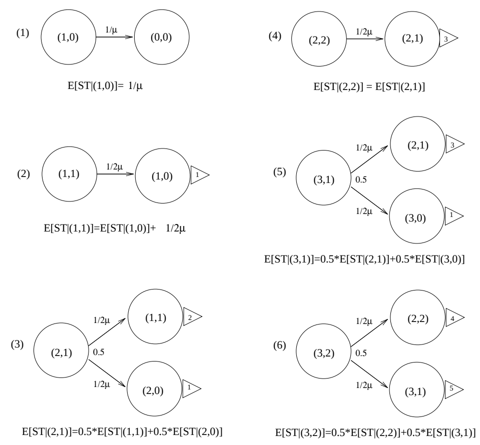

Figure 6: ET-S model of mean service time of a job at the parallel subsystem

Theorem 6.2 $E[ST|(n,n)^+] = E[ST|(n,n-1)^+]$ $\forall n > 1$, *and* $E[ST|(n_1,n_2)^+] = \frac{1}{2}E[ST|(n_1-1,n_2)^+] + \frac{1}{2}E[ST|(n_1,n_2-1)^+]$, *where wlog, assume* $n_1 > n_2 \geq 1$.
The basis equations for the recursion are given by, $E[ST|(1,0)^+] = \frac{1}{\mu} = \frac{H_1}{\mu}$, *and* $E[ST|(1,1)^+] = \frac{1}{2\mu} + E[ST|(1,0)^+] = \frac{3}{2\mu} = \frac{H_2}{\mu}$.

Proof: Figure 6 illustrates the proof using recursive ET-S trees. The roots of each tree (except the atom (1), which represents $(1,0)^+$) represent the state seen just after arrival of the tagged job whose pservice time is to be calculated. Note that, $E[ST|(n,0)^+] = E[ST|(1,0)^+] = \frac{1}{\mu}$. □

The following three corollaries follow directly from Theorem 6.2.

Corollary 6.4 $E[ST|(n,0)^+] < E[ST|(n,1)^+] < \cdots < E[ST|(n,n-1)^+] = E[ST|(n,n)^+]$

Corollary 6.5 *If* $n > s$ *then* $E[ST|(n,n)] < E[ST|(s,s)]$

Corollary 6.6 *If* $n > s$, *then* $E[ST|(n,s)^+] < E[ST|(n-1,s)^+] < \cdots < E[ST|(s+1,s)^+] = E[ST|(s,s)^+]$

7 Pwait Time Analysis

The pwait time of a job is equal to the pservice times of all the jobs ahead of it in the queue. Therefore, if $S^{a+} = (n,1)$, $n > 1$, (i.e., all the (n-1) jobs ahead of the tagged job have a task waiting at the join point) then, $E[WT|(n,1)^+] = \sum_{i=1}^{n-1} E[ST|(n-i,0)^+] = (n-1) \times \frac{1}{\mu}$.

If $S^{a+} = (n,n)$, $n > 1$, then, $E[WT|(n,n)^+] = \sum_{i=1}^{n-1} E[ST|(n-i,n-i)^+]$.
If $S^{a+} = (n_1,n_2)$, with $n_1 > n_2 > 1$ then, $E[WT|(n_1,n_2)^+] = \sum_{i=1}^{n_2-1} E[ST|(n_1-i,n_2-i)^+] + (n_1-n_2)\frac{1}{\mu}$.
Thus, unlike non-parallel systems, knowledge of just the number of jobs waiting ahead of a tagged job does not provide enough information to calculate the mean wait time of the job. It is necessary to know the specific state of the parallel subsystem at the instant of arrival. This is because mean pservice time of a tagged job in a parallel subsystem is arrival state dependent, whereas, mean service time of a job in a serial subsystem is a constant. The pwait time of an incoming job that sees the system in state $(n_1,n_2)^{a+}$, $n_1 \geq n_2 \geq 1$, with $N(= n_1 - 1)$ jobs ahead of it in the parallel subsystem is equal to the sum of random variables $(X_1, X_2, \cdots, X_N)$; where the X_i's are the pservice times of the N earlier jobs, which are dependent on the specific state seen by the tagged job at arrival time.

8 Conclusions

The mean response time of a job at a fork-join system is shown to be the sum of pwait and pservice times which are analogous to wait and service times in non-parallel systems. It is proved that pservice time of a job at the fork-join subsystem is dependent on the state at arrival time of the job. In particular, the mean pservice time of a job decreases as: (a) the number of jobs at the join point increases, and (b) number of jobs in the parallel subsystem increases. It is also shown that when pservice time includes only the execution time of a single task, the service time of its sibling task vanishes from the mean response time analysis, "speeding up" the system (i.e., mean time spent by a job at the synchronization point decreases). A formal pwait time analysis, similar to the wait time analysis of non-parallel systems, is also given. The analysis presented in this paper is carried out using a simple modeling technique called ET-S trees, which is introduced in this paper.

Acknowledgements

We thank Cynthia Childers for her comments which have greatly improved this paper.

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

[3] Sevcik, K. C., Mitrani, I. "The distribution of queueing network states at input and output instants", Journal of the Association of Computing Machinery, vol. 28, no. 2, April 1981, pp. 358 – 371.

[4] Varki, E., Dowdy, L. W. "Exact response time analysis of two server fork-join systems", Technical Report, Vanderbilt University, 1995.