On Maximizing the Number of Departures Before a Deadline on Multiple Processors

Randolph Nelson
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598

Don Towsley
Department of Computer and Information Science
University of Massachusetts
Amherst, MA 01003

COINS Technical Report 87-99
July 15, 1987

On Maximizing the Number of Departures Before a Deadline on Multiple Processors

Randolph Nelson
IBM T. J. Watson Research Center
P. O. Box 218
Yorktown Heights, NY 10598

Don Towsley [1]
Department of Computer and
Information Science
University of Massachusetts
Amherst, MA 01003

Abstract.

Consider a set of processors $P_1, P_2, ..., P_N$ differing only in speed, a set of M jobs with exponentially distributed execution times, and an exponentially distributed deadline. The problem we consider is to schedule the jobs non-preemptively so as to maximize the expected number of job completions before the deadline. We prove that a threshold policy achieves this objective and provide an algorithm to calculate the values of the thresholds.

In this paper we consider a system of multiple processors, P_1, P_2, ..., P_N, where the servicing rates of the processors are, in general, not identical, serving a collection of $M > 0$ identical jobs. Associated with this system is an exponentially distributed deadline by which time the system shuts down. We are concerned with determining a policy by which jobs are non-preemptively scheduled on available free processors so as to maximize the expected number of job completions prior to system shutdown. Our main result shows that a threshold scheduling policy maximizes this objective. Moreover we provide an efficient algorithm that calculates these thresholds. In a threshold

[1] This work was performed while D. Towsley was a visiting scientist at the IBM Thomas J. Watson Research Center. This work was also supported in part by the National Science Foundation under grant ECS-8406402

policy there exist integer valued thresholds T_i, with $1 = T_1 \leq T_2 \leq \ldots \leq T_N < \infty$ such that the policy schedules an available job to an idle processor P_i only if T_i is smaller or equal to the threshold of any other idle processor *and* if the number of available jobs is greater or equal to T_i.

Systems of this type could be models for reliability of computer systems, multiprocessor scheduling, or other environments where tasks have to be completed by a deadline. The problem in a multiprocessor environment containing different speed processors is to minimize the expected delay of jobs. The structure of the optimal policy for two processors has been shown to be a threshold policy, Lin and Kumar (1984), however determining the optimal policy for three or more processors appears to be very difficult. A heuristic for minimizing the mean response time is to maximize the number of job completions between successive arrivals to the system. Within the context of our model, then, a deadline corresponds to a job arrival.

Agrawala et al. (1984)., determined an optimum policy for minimizing the expected total flow time (the sum of finishing times) of a set of jobs on a a multiple processor system. They show that the optimum policy is a threshold policy and obtain an explicit expression for the thresholds. Our work differs from theirs in that we seek policies that maximize the expected throughput on a multiprocessor system prior to some deadline. The thresholds obtained in for this objective are, in general, different than those found in Agrawala et al. (1984). Other work concerning a fixed number of jobs is focused on minimizing the total time needed to process all the jobs on multiple processors where preemptions are allowed Weiss and Pinedo (1980) and Weber (1982).

Models which consider multiple processor systems where arrivals are allowed have also been studied. In this case the objective is to minimize the expected response time of a job. In Larsen and Agrawala (1983) a two server system is considered and the value of queue length after which one should schedule the slower server, assuming a threshold policy, is calculated. In Lin and Kumar (1984) it is shown that for a two server system a threshold policy minimizes the expected

response time. A simpler proof was developed by Walrand (1984). Kumar and Walrand (1985)explored the relationship between individual optimal and socially optimal policies. They gave conditions under which a socially optimal policy for a system with no arrivals is individually optimal with or without arrivals and used this result to show that threshold policy of Agrawala et al. (1984). is individually optimal. This results can be used to show that our policy is individual optimal in the case of no arrivals. In Chow and Kohler (1979) various scheduling policies are considered for a multi-queue system where the servers have different speeds. In this work they study a scheduling discipline that is concerned with maximizing the expected number of departures between successive arrivals. A number of papers have considered exact or approximate analysis of the expected response time for heterogeneous multiprocessor systems with threshold scheduling. In Cooper (1976) exact results are derived for policies that schedule the fastest available server. Rubinovitch (1985) derived exact expressions for mean response time for two processor systems and Nelson and Towsley (1984) derived an approximation for the mean response time for a arbitrary number of processors.

In Section 1 we formalize our problem and provide the necessary definitions and notation. In Section 2 we prove that a threshold policy maximizes the number of job completions before the deadline and provide an algorithm for calculating the thresholds. In Section 3 we compare the performance of the derived threshold policy used an a heuristic for minimizing the expected response time to other threshold policies and finally in Section 4 we suggest directions for future research.

1. Definitions and Notation

We consider a set of N heterogeneous processors $\{P_1, P_2, ..., P_N\}$ that serve a collection of M jobs. The service time for a job executed on processor P_i is an exponentially distributed random variable with mean $1/\mu_i$, $i = 1, ...,N$ respectively. We assume that the processors are ordered so that $\mu_1 \geq \mu_2 \geq ... \geq \mu_N > 0$. We are interested in maximizing the number of job completions by a deadline that is assumed to be an exponentially distributed random variable with mean $1/\lambda$.

We define a threshold policy by associating with each processor, P_i, a positive integer valued threshold, T_i. A job from the queue is scheduled to an idle server P_i only if T_i is smaller than or equal to the threshold of any other idle server and if the queue length is greater than or equal to T_i. For example, consider $N = 2$ and $T_1 = 1$ and $T_2 = 4$. Then processor 2 will only be scheduled if the queue length is greater than or equal to 4.

Let (m,c) denote the state of the system where m represents the number of jobs waiting to be processed and $c = (c_1, c_2, ..., c_N)$ where $c_i = 1$ if the ith processor P_i is busy and $c_i = 0$ otherwise. We shall refer to c as the *configuration* of the system. We define e_k to be the configuration in which only the kth processor is busy and v_k to be the configuration in which only the fastest k processors are busy. We let v_0 be the configuration in which there are no busy processors. Let the set of busy processors be given by $B(c) = \{i \mid c_i = 1\}$, the number of busy processors by $|c| = c_1 + c_2 + ... + c_N$, and the number of jobs in the system by $|(m,c)| = m + |c|$. Last, we define two operations on system configurations. If c and c' are two configurations, then the busy processors in either or both of them is given by the sum, $d = c + c'$, where $d_i = \min\{1, c_i + c'_i\}$, $i = 1,...,N$. The number of processors that are busy in both configurations is given by the inner product , $c \bullet c'$, and satisfies $\sum_{i=1}^{N} c_i c'_i$.

Because of the exponential assumptions, scheduling decisions can be limited to system startup time and to points in time where a processor completes a job. A scheduling policy f maps system states into system states. We let F denote the set of all scheduling policies. A state (m, c) is *stable* with respect to policy f if $f(m, c) = (m, c)$, i.e., no processors are immediately scheduled when f is applied to state (m, c) otherwise we say that state (m, c) is *unstable* with respect to f. A scheduling policy $f \in F$ satisfies the following properties: if $f(m, c) = (n, d)$, then

1. $d_k \geq c_k$, $k = 1, ..., N$ and $|(m, c)| = |(n, d)|$,
2. (n, d) is stable with respect to f.

The policy $f(m, c) = (n, d)$ can be interpreted as the assignment of $m - n$ jobs to those $m - n$ processors P_k such that $c_k = 0$ and $d_k = 1$.

Assume the system starts in state (m,c). We define $E^f(m, c)$ to be the average number of job completions before the deadline for an initial state (m, c), when one uses policy f beginning at the *first* departure. If we apply policy f initially, then the expected number of job completions before the deadline is $E^f(f(m,c))$. It is clear that $E^f(m, c)$ is not necessarily equal to $E^f(f(m, c))$. Our objective is given by

$$E(m, c) = \max_{f \in F} E^f(f(m, c)). \tag{1}$$

Any policy f for which $E^f(m, c) = E(m, c)$ for all (m, c) is an optimal policy. We let F^* be the set of all optimal policies. Using a renewal argument, we can write

$$E^f(m, c) = 1/(\lambda + U(c)) \sum_{r \in B(c)} \mu_r [1 + E^f(f(m, c - e_r))]$$

where we define

$$U(c) = \sum_{r \in B(c)} \mu_r.$$

A policy is a *single item* (SI) policy if it never schedules more than one processor at any one time. We let G be the set of all single item scheduling policies and define $S(m, c) = \max_{g \in G} E^g(m, c)$. We say that policy $g \in G$ is an optimal single item policy if $E^g(m, c) = S(m, c)$ for all (m, c).

For a state (m, c) where $|(m, c)| > 0$ we define $i^f(m, c)$ to be the index of the slowest processor that might be scheduled before the deadline after initially applying f and for all possible departures using policy f from state (m, c). It is clear that processors with an index higher than $i^f(m, c)$ will never be scheduled a job under policy f for an initial state (m, c). We let $i(m, c) = \min_{f \in F^*} \{i^f(m, c)\}$. and will only consider optimal policies f that satisfy $i^f(m, c) = i(m, c)$. We next define $h(m) = i(m, v_N)$, $m > 0$, and $h(0) = 0$. It is clear that $h(m)$ is the slowest processor that will be scheduled by an optimal policy given an initial state of (m, v_N). Finally we define integer values, T_j, given by:

$$T_j = m \text{ if } h(m-1) < j \le h(m),\ j = 1, 2, \ldots, N,\ m \ge 1. \tag{2}$$

Theorem 1 of Section 2 shows that the optimal policy is a threshold policy with these thresholds.

2. Analysis

In this section we establish that a threshold policy maximizes the number of job completions before the deadline. We first establish two properties for optimal policies. These properties are:

1. **(FAP Property)** There exists an optimal policy such that whenever it schedules, it always schedules jobs to the fastest available processor.
2. **(IP Property)** After the first instance that a processor is not scheduled after a departure, it remains idle until the deadline.

These properties are used to establish that there exist a optimal policy that is a threshold policy. We now prove two lemmas that establish these properties.

Lemma 1. (FAP Property)

If two configurations c and c' satisfy the relation $c \bullet v_k \geq c' \bullet v_k$ for $k = 1, ..., N$ and $|c| = |c'|$, then for any optimal policy f and any optimal SI policy

a. $E(f(m, c)) \geq E(f(m, c'))$ $m = 0,1, ...$

b. $S(m, c) \geq S(m, c')$ $m = 0,1, ...$

Proof (Special Case).

We shall prove Lemma 1.a first for a special configuration which can be extended to the general case. In the special case we assume that $c_k = 1$, $c_j = 0$, and $c' = c - e_k + e_j$, $j > k$. We prove the special case by induction on $|(m, c)|$.

Basis step (Special Case).

For $|(m, c)| = 1$, $E(f(0, e_k)) = E(0, e_k) \geq E(0, e_j) = E(f(0, e_j))$ $j \geq k$.

Inductive Step (Special Case).

Assume that Lemma 1.a is true for the special case for $|(m, c)| = a$ and consider a state (m, c) such that $|(m, c)| = a + 1$. Let f be an optimal policy. We construct a policy f' that operates on (m, c) in the following manner: if $f(m, c')$ schedules P_i, $i \neq k$, then $f'(m, c)$ schedules P_i, if $f(m, c')$ schedules P_k, then $f'(m, c)$ schedules P_j, otherwise nothing else is scheduled. Since f is optimal we have $E(f(m, c)) \geq E(f'(m, c))$. We will now show that $E(f'(m, c)) \geq E(f(m, c'))$ and thus that $E(f(m, c)) \geq E(f(m, c'))$. This is certainly true if $f(m, c')$ schedules P_k since for this case $f'(m, c) = f(m, c')$. Assume then that $f(m, c')$ does not schedule P_k. Let $f(m, c') = (n, d - e_k + e_j)$ and thus, by definition of policy f', $f'(m, c) = (n, d)$ and let $D = B(d) - \{k\}$. We write expressions for $E(n, d)$ and $E(n, d - e_k + e_j)$:

$$E(n, d) = \frac{1}{\lambda + U(d - e_k) + \mu_k} \sum_{i \in D} \mu_i \, [1 + E(f(n, d - e_i))] + \frac{\mu_k}{\lambda + U(d - e_k) + \mu_k} [1 + E(f(n, d - e_k))],$$

$$E(n, d - e_k + e_j) = \frac{1}{\lambda + U(d - e_k) + \mu_j} \sum_{i \in D} \mu_i \, [1 + E(f(n, d - e_i - e_k + e_j))]$$

$$+ \frac{\mu_j}{\lambda + U(d - e_k) + \mu_j} [1 + E(f(n, d - e_k))].$$

By induction, $E(f(n, d - e_i)) \geq E(f(n, d - e_k + e_j - e_i))$. Therefore,

$$E(n, d) - E(n, d - e_k + e_j) \geq \sum_{i \in D} \mu_i (\mu_j - \mu_k) \frac{[1 + E(f(n, d - e_i)]}{(\lambda + U(d - e_k) + \mu_k)(\lambda + U(d - e_k) + \mu_j)}$$

$$- \frac{(\mu_j - \mu_k)[1 + E(f(n, d - e_k))](\lambda + U(d - e_k))}{(\lambda + U(d - e_k) + \mu_k)(\lambda + U(d - e_k) + \mu_j)},$$

$$\geq \frac{(\mu_j - \mu_k)}{(\lambda + U(d - e_k) + \mu_j)} \frac{1}{(\lambda + U(d - e_k) + \mu_k)} \sum_{i \in B(d)} \mu_i \, [1 + E(f(n, d - e_i))]$$

$$- \frac{\mu_j - \mu_k}{\lambda + U(d - e_k) + \mu_j} [1 + E(f(n,\ d - e_k)]$$

$$\geq \frac{(\mu_j - \mu_k)}{(\lambda + U(d - e_k) + \mu_j)} [E(n, d) - 1 - E(f(n, d - e_k))]$$

$$\geq 0,$$

where the last inequality follows from the fact that $E(n, d) \leq E(f(n, d - e_k)) + 1$.

This shows that $E(f'(m,\ c)) \geq E(f(m,\ c'))$ and completes the proof for the special case.

Proof (General Case).

We now generalize this result to any c and c'. For any (m, c) and (m, c') satisfying the conditions of Lemma 1, there exists a sequence of configurations $c = z_0, z_1, \ldots, z_r = c'$ such that the pairs $(z_i,\ z_{i+1}), 0 \leq i \leq r$ each differ in exactly two processor assignments and $z_i \cdot v_k \geq z_{i+1} \cdot v_k$ for $1 \leq k \leq N$. Hence the proof for the special case provides

$$E(f(m, c)) \geq E(f(m, z_1)) \geq \ldots \geq E(f(m, z_{r-1})) \geq E(f(m, c'))$$

which proves Lemma 1.a. The proof for Lemma 1.b is similar and is thus omitted.

■

Lemma 2. (IP Property)

Let f be an optimal policy and (n, d) be a stable state with respect to f. If for any j, P_j is idle in (n, d) then P_j is idle in $f(n,\ d - e_i)$ for all $i \in B(d)$.

Proof

This lemma is trivially true for $j > i(n, d)$. Let (n, d) be a state having the property that $|(n,\ d)|$ is minimal and violates the statement of the lemma, i.e. there exists a j such that $d_j = 0$ and there

exists an $i \in B(d)$ such that P_j is scheduled in $f(n,\ d - e_i)$. Furthermore let j^* be the minimal j that has this property. We first show that $P_j{}^*$ is scheduled in $f(n,\ d - e_k)$ for all $k \in B(d)$. To do this suppose that $P_j{}^*$ is not scheduled in $f(n,\ d - e_i{}^*)$, $i^* \in B(d)$. We will show that this leads to a contradiction. There are two cases:

Case 1. State $(n,\ d - e_i{}^*)$ is stable with respect to f.

We show that this implies that $\lambda(n, d - e_i{}^*) = \lambda(n,\ d)$ which leads to a contradiction since, from the definition of $\lambda()$, it implies that there exists a set of possible departures from state $(n,\ d - e_i{}^*)$ such that processor $P_j{}^*$ is scheduled and contradicts the minimality of $|(n,\ d)|$. It is clear, since state $(n,\ d - e_i{}^*)$ arises from a departure from state $(n,\ d)$ that $\lambda(n,\ d) \geq \lambda(n,\ d - e_i{}^*)$. To show that $\lambda(n,\ d) \leq \lambda(n,\ d - e_i{}^*)$ and thus establish equality, we show that the following equation holds:

$$E(n, d) = E(n, d - e_i{}^*) + \frac{\mu_i{}^*}{\lambda + \mu_i{}^*}. \tag{3}$$

This implies that there exists a optimal policy which operates by performing identical scheduling decisions as optimal policy acting on $(n,\ d - e_i{}^*)$ while ignoring processor $P_i{}^*$. As this optimal policy might not satisfy the definition of $\lambda()$ this implies that $\lambda(n,\ d) \leq \lambda(n, d - e_i{}^*)$. To establish the equality we first show that

$$E(n, d) \geq E(n,\ d - e_i{}^*) + \frac{\mu_i{}^*}{\lambda + \mu_i{}^*}. \tag{4}$$

To establish this we consider a possibly suboptimal policy, f', that simulates the scheduling decisions of f from state $(n,\ d)$ but ignores processor $P_i{}^*$. It is clear that f from state $(n,\ d - e_i{}^*)$ never schedules processor $P_i{}^*$ since this would violate the minimality of $|(n,\ d)|$. Thus f' will never schedule $P_i{}^*$ for all possible departures from state $(n,\ d)$. Equation (4) follows from the fact that $E(n, d)$ must be at least as large as $E^{f'}(n,\ d)$. To show the reverse inequality we consider a possibly suboptimal policy that adds a fictitious job to processor $P_i{}^*$ and schedules jobs in the same fashion

as the optimal policy from state (n, d). The throughput of the fictitious job is equal to $\mu_i^*/\lambda + \mu_i^*$ and since this policy is possibly suboptimal this implies that

$$E(n, d - e_i^*) \geq E(n, d) - \frac{\mu_i^*}{\lambda + \mu_i^*}.$$

which provides the opposite inequality and establishes equation (3).

Case 2. State $(n, d - e_i^*)$ is not stable with respect to f. There are two subcases:

Case 2a. $i^* > j^*$.

Since $(n, d - e_i^*)$ is not stable there exists a processor P_k, $k \leq i(n, d - e_i^*) \leq i(n, d)$, that is scheduled. By assumption and the FAP property it must be the fact that $k < j^*$ but this implies that $d_k = 0$ in (n, d) and that P_k is scheduled in $f(n, d - e_i^*)$ which contradicts the minimality of j^*.

Case 2b. $i^* < j^*$.

This implies, by the FAP property, that only P_i^* is scheduled since scheduling a processor $k < i^* < j^*$ would contradict the minimality of j^*. This implies that state $(n - 1, d)$ is stable and that

$$E(n - 1, d) > E(n, d - e_i^*). \tag{5}$$

We now show that $(n - 1, d)$ cannot be stable and thus reach a contradiction that processor P_j^* is not scheduled. To see this consider an $i \in B(d)$ such that P_j^* is scheduled in $f(n, d - e_i)$. By assumption such an i must exist. There are two cases for scheduling processors from state $(n, d - e_i)$, namely

Case i. Schedule processor P_j^* and processors P_j, $j \in J$, where $J = \{j \mid j \neq j^*, j \neq i, P_j$ is scheduled from state $(n, d - e_i)\}$ but do not schedule processor P_i. We let c be defined so that $c_j = 1$, $j \in J$ and zero otherwise.

Case ii. Schedule processors P_{j^*} and and processors P_j, $j \in J'$, where $J' = \{j \mid j \neq j^*, j \neq i, P_j \text{ is scheduled from state } (n, d - e_i)\}$ and also schedule processor P_i. We let c' be defined so that $c'_j = 1$, $j \in J'$ and zero otherwise.

In case i it is clear from the FAP property that $i > j^*$ and from Lemma 1 that the following equation holds:

$$E(n-1-|c|, d-e_i+e_{j^*}+c) \geq E(n-1, d-e_i+e_{j^*}) \geq E(n-1, d). \tag{6}$$

From case ii we have the following equation:

$$E(n-2-|c'|, d+e_{j^*}+c') \geq E(n-1, d). \tag{7}$$

Comparing equation (5) with equations (6) and (7) shows that $(n-1, d)$ cannot be stable since there exists a state that can be obtained by scheduling other processors from state $(n-1, d)$ that increases the expected number of departures before the deadline.

We thus conclude that processor P_{j^*} must be scheduled in $f(n, d-e_i)$ for all $i \in B(d)$. We continue the proof by defining a new policy f' which schedules P_{j^*} at the beginning, i.e. $f'(n, d) = (n-1, d+e_{j^*})$, and then continues as in policy f, i.e. $f'(n-1, d+e_{j^*}-e_i)$, $i \in B(d)$ is the same as $f(n, d-e_i)$ except that P_{j^*} is already scheduled. There are two cases to consider depending upon if processor P_{j^*} in state $(n-1, d+e_{j^*})$ is the first processor to complete or not. In the first case we assume that P_{j^*} is the first processor to complete. For this case the expected number of completions before the deadline is given by $E(n-1, d)+1$ which is greater than or equal to $E(n, d)$. In the second case we assume that processor P_{j^*} is not the first processor to complete. In this case the expected number of completions before the deadline is $E(n, d)$. Thus the expected number of completions before the deadline cannot be decreased by scheduling P_{j^*}, and may even increase. Therefore we reach a contradiction and the lemma is proved.

■

We make several remarks that follow directly from the previous lemmas. These will be used in proving the theorems stated later in the paper. As their proofs are easy we omit the details.

Remarks.

1. There exists a *unique* optimal policy π such that $i^{\pi}(n, d) = i(n, d)$. Henceforth we restrict our attention to policy π, and when we say a state is stable (unstable) we mean with respect to this policy.
2. A state (n, d) is stable if and only if P_j is busy for all $j \leq i(n, d)$.
3. The following equality holds: $h(m) = i(m, v_{h(m)})$.

We next establish a lemma that characterizes the function $h(m)$.

Lemma 3.

The function $h(m)$ is a non-decreasing function of m.

Proof.

We prove this by contradiction. Assume that $h(n) > h(n+1)$, $n > 0$. Consider state $(n+1, v_{h(n+1)})$. A departure from processor P_1 from this state, from the FAP property, leads to state $(n, v_{h(n+1)})$ which, since we are assuming $h(n) > h(n+1)$, implies, from remark 2 and FAP, that the optimal policy will schedule processor $P_{h(n+1)+1}$. This implies that $h(n+1) > i(n+1, v_{h(n+1)})$ and contradicts remark 3.

■

We are now in a position to show that the optimal policy π is a threshold policy.

Theorem 1.

The optimal policy π is a threshold policy in which the threshold for processor P_i is given by T_i defined in equation (2).

Proof (Stable State).

Our proof consists in showing that π and the threshold policy having thresholds given by equation (2) perform identical scheduling operations. We first show that the policies are identical for stable states and then show that this is also true for unstable states. Assume that (m, c) is stable with respect to π. This implies that $c_k = 1$ for $k \leq h(m)$. We apply the threshold policy to state (m, c). Let P_j be the idle processor with the lowest threshold. Since, from Lemma 3, $h(m)$ is a non-decreasing function of m, it follows from equation (2) that T_j is a non-decreasing function of j, $j > h(m)$. Processor P_j will only be scheduled if $T_j \leq m$. According to equation (2) this implies that $j \leq h(m)$ which is a contradiction. Therefore no processor will be scheduled and the threshold policy and π make identical decisions.

Proof (Unstable States).

Assume (m, c) is unstable. The policy π schedules r processors $P_{i_1}, P_{i_2}, ..., P_{i_r}$, $i_1 < i_2 < ... < i_r$ resulting in the stable state $(m-r,\ c')$ where $c'_k = 1$ if $c_k = 1$ or if $k = i_j$, $j = 1, 2, ..., r$. It must be the case that $h(m-j+1) \geq i_j$, $j = 1, 2, ..., r$. Consider the application of the threshold policy on state (m, c). We claim that it will be applied r times and result in the scheduling of the same r processors in the order of increasing index values. We consider the jth application of the threshold policy. This application will be on the state $(m-j+1, d)$ where $d_k = 1$ if $c_k = 1$ or $k = i_1, i_2, ..., i_{j-1}$. The idle processor with the lowest threshold is P_{i_j}. As we know that $h(m-j+1) > i_j$, it must be the case from equation (2) that $T_{i_j} \leq m-j+1$. Consequently P_{i_j} will be scheduled at this time. The rth application results in the state $(m-r, c')$ which is stable. Consequently the threshold policy will not schedule another processor and thus both the threshold policy and π make identical scheduling decisions.

■

We are interested now in providing an algorithm with which to calculate the thresholds. First we establish how the optimal policy and the function $h(m)$ is related to the optimal SI policy.

Lemma 4.

The optimal policy π starting from a stable state (m, c) is a SI policy.

Proof.

Assume that $P_j, j \in B(c)$ is the first processor to complete. If $j > h(m)$ then no processor will be scheduled. If $j \leq h(m)$ then P_j will be rescheduled resulting in a state $(m-1, c)$. Since, from Lemma 3, $h(m-1) \leq h(m)$ there cannot be any other processor scheduled and thus at most one processor is scheduled.

■

Theorem 2.

The function $h(m)$ has the following relationship with $S(m, c)$,

$$h(m) = \min\{k \mid S(m-1, v_{k+1}) \leq S(m, v_k)\}, m = 1,2, \ldots \tag{8}$$

Proof.

We prove this theorem by contradiction. Assume that $m = n$ is the *smallest value* of m that violates equation (8). There are two cases:

Case 1. $S(n-1, v_{h(n)+1}) > S(n, v_{h(n)})$. (9)

From remark 3 we know that $(n, v_{h(n)})$ is a stable state. Consequently, from Lemma 4, π is an SI policy when applied to $(n, v_{h(n)})$ and $E(\pi(n, v_{h(n)})) = S(n, v_{h(n)})$. However equation (9) shows that the throughput is improved by scheduling $P_{h(n)+1}$. This contradicts the fact that $(n, v_{h(n)})$ is stable and therefore $S(n-1, v_{h(n)+1}) \leq S(n, v_{h(n)})$.

Case 2. $S(n-1, v_{h(n)+1}) \le S(n, v_{h(n)})$ and there exists a $k < h(n)$ such that $S(n-1, v_{k+1}) \le S(n, v_k)$.

Let k be the minimal processor index value so that Case 2 is true. We first suppose that $k < h(n-1)$. We will show that this leads to a contradiction. We have the following expressions for $S(n-1, v_{k+1})$ and $S(n, v_k)$.

$$S(n-1, v_{k+1}) = \frac{1}{(\lambda + U_{k+1})} \sum_{r=1}^{k+1} \mu_r \left[1 + \max\{S(n-2, v_{k+1}), S(n-1, v_{k+1} - e_r)\}\right], \quad (10)$$

$$S(n, v_k) = \frac{1}{(\lambda + U_k)} \sum_{r=1}^{k} \mu_r \left[1 + \max\{S(n-1, v_k), S(n, v_k - e_r)\}\right]. \quad (11)$$

By our choice of k we know that $S(n-1, v_k) \ge S(n, v_{k-1})$ and, by Lemma 1.b, that $S(n, v_{k-1}) \ge S(n, v_k - e_r)$ for $1 \le r \le k-1$. Hence $S(n-1, v_k) \ge S(n, v_k - e_r)$ for $1 \le r \le k-1$. Using this we can rewrite equation (11) as

$$S(n, v_k) = \frac{U_k}{(\lambda + U_k)} \left[1 + S(n-1, v_k)\right]. \quad (12)$$

By our choice of n and because $k < h(n-1)$, we have $S(n-1, v_k) \le S(n-2, v_{k+1})$. Using this in equation (12) leads to

$$S(n, v_k) \le \frac{U_k}{(\lambda + U_k)} \left[1 + S(n-2, v_{k+1})\right]. \quad (13)$$

The following relationship is easily shown. $U_{k+1}/(\lambda + U_{k+1}) \ge U_k/(\lambda + U_k)$ Hence, since $S(n-2, v_{k+1}) \le \max\{S(n-2, v_{k+1}), S(n-1, v_{k+1} - e_r)\}$, equations (13) and (10) yield

$$S(n, v_k) \le \frac{U_{k+1}}{(\lambda + U_{k+1})} \left[1 + S(n-2, v_{k+1})\right] \le S(n-1, v_{k+1})$$

which is a contradiction that $k < h(n-1)$. Therefore it must be the case that $k \ge h(n-1)$. Since $k < h(n)$ it must be the case that (n, v_k) is unstable. If we apply π as a threshold policy to state (n, v_k) we will schedule P_{k+1} first. The resulting state is $(n-1, v_{k+1})$ which, because

$k+1 > h(n-1)$, is stable. This implies that $S(n, v_k) \leq S(n-1, v_{k+1})$ which contradicts the assumption of Case 2. Therefore it must be the case that equation (8) is true.

■

As a result of Theorem 2, the following algorithm can be used to obtain the thresholds T_j, $j = 1, 2, ..., N$.

Algorithm

1. Initialize.

   $T_1 := 1$

   $k := 1$

   $m := 0$

2. Main Loop.

   while $k \neq N$ do

   $m := m + 1$

   while $S(m, v_k) < S(m-1, v_{k+1})$ do

   $k := k + 1$

   $T_k := m$

3. Stop.

The function $S(m, v_k)$ is given by the recursive expression, $S(0, v_k) = \sum_{r=1}^{k} \mu_r/(\mu_r + \lambda)$, $S(m, v_0) = 0$, and for $m \geq 1, k \geq 1$,

$$S(m, v_k) = \frac{1}{\lambda + U(v_k)} \sum_{r=1}^{k} \mu_r (1 + \max\{S(m-1, v_k), S(m, v_{r-1}) + \sum_{j=r-1}^{k} \frac{\mu_j}{\mu_j + \lambda}\})$$

Define m_k to be the smallest integer such that $h(m_k) \geq k$ and $h(m) < k$ for $m < m_k$. The following theorem shows that $m_k < \infty$, for $k = 1, 2, ..., N$. As a consequence, the preceding algorithm is guaranteed to halt in a finite number of steps. Last, although not described here, we have im-

plemented the above algorithm in such a way that the number of states (m, v_k) that we must consider is proportional to m_N and independent of N.

Theorem 3.

For each processor P_k, $k = 1, ..., N$ there exists an integer $m_k < \infty$ such that $h(m_k) \geq k$ and $h(m) < k$, $m < m_k$.

Proof.

We prove the theorem by contradiction. Assume that P_k, $k < N$, is the slowest processor satisfying $h(m_k) = k$, $m_k < \infty$ and that processor P_{k+1} is never scheduled, $h(m) = k$, $m \geq m_k$. From the FAP property this implies that the optimal policy, π, never schedules processors P_{k+j}, $j = 1, 2, ..., N - k$. Let f^{k+1} be a threshold policy with thresholds $T_i = 1, i \leq k + 1$, and $T_i = \infty, i > k + 1$. Our proof consists in showing that

$$\lim_{m \to \infty} (E^{f^{k+1}}(m, v_k) - E(m, v_k)) > 0 \tag{14}$$

which contradicts the optimality of π and shows that processor P_{k+1} must be scheduled in the optimal policy.

To show this we first establish an upper bound on $E(m, v_k)$. Clearly, since by assumption π does not schedule processors j, $j > k$, this system cannot process as many jobs before the deadline as a single processor system with a processing rate $\sum_{i=1}^{k} \mu_i$. Therefore,

$$E(m, v_k) \leq U(v_k)/\lambda \tag{15}$$

We next show that

$$\lim_{m\to\infty} E^{f^{k+1}}(m,\ v_k) \geq U(v_{k+1})/\lambda \tag{16}$$

which establishes equation (14) since $U(v_k) < U(v_{k+1})$. To show equation (16) let $q = U(v_{k+1})/(\lambda + U(v_{k+1}))$ and define $p_i(m)$ to be the probability that there are i departures before the deadline when using policy f^{k+1} on an initial state $(m,\ v_k)$. It is clear that $p_i(m) = q^i(1-q),\ \ i < m$. The number of departures before the deadline for policy f^{k+1} from state $(m,\ v_k)$ satisfies the following

$$E^{f^{k+1}}(m,\ v_k) = \sum_{i=0}^{m+k} i p_i(m) > \sum_{i=0}^{m-1} i p_i(m) \quad m > 0 \tag{17}$$

Equation (16) follows by taking the limit as $m \to \infty$ in equation (17).

■

3. Comparison of Threshold Policies

In this section we compare three different threshold policies to determine their performance characteristics. We use these policies as heuristics for minimizing the expected job response time with Poisson arrivals. The three policies we consider are

1. Fastest Available Server (**FAS**) Policy - In this policy, each threshold has value one, $T_i = 1,\ i = 1, ..., N$. In the case where there are two or more available processors, jobs are scheduled to the fastest processor. The performance of this policy was studied in Cooper (1976).

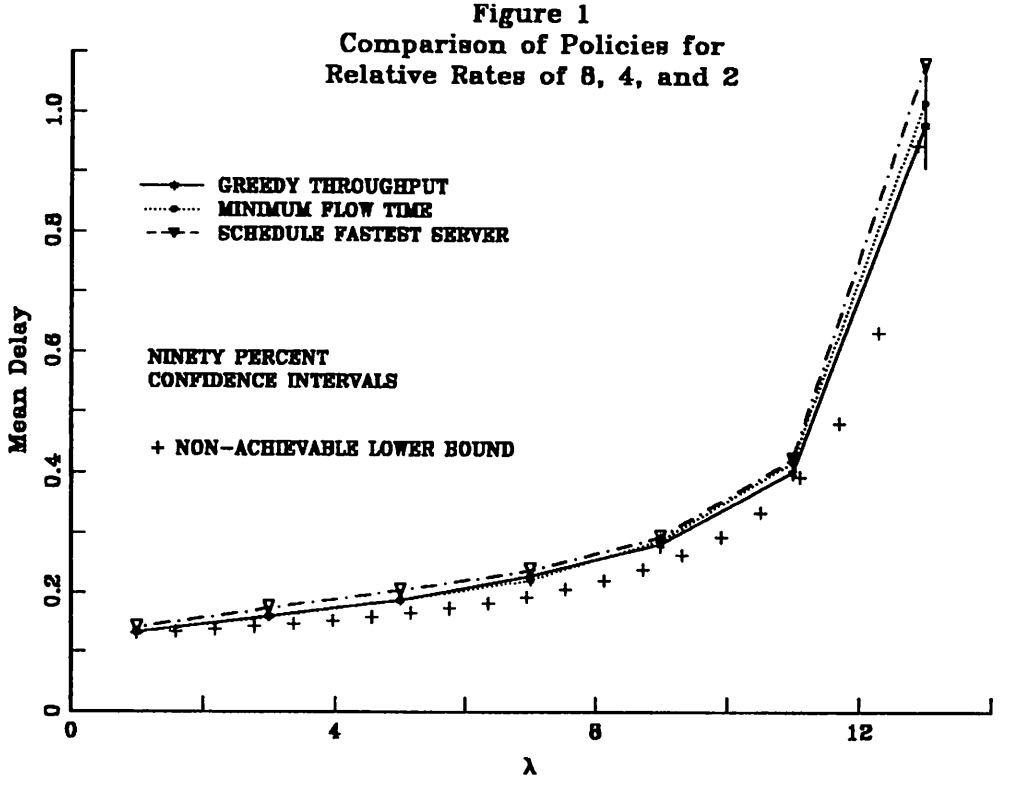
2. Minimum Flow Time (**MFT**) Policy - This policy was shown to minimize the expected flow time for a collection of jobs on N servers Agrawala et al. (1984). We shall use this as a heuristic for the N server system with arrivals. The i^{th} threshold, T_i, is given by the least integer that is greater than or equal to $T_i = (\sum_{r=0}^{i-1} \mu_r)/\mu_i - (i-1),\ i = 1, ..., N$.

3. Greedy Throughput (**GT**) Policy - This is the policy derived in Section 2 and thus maximizes the mean number of job completions from the system prior to an exponential deadline where, for this case, the deadline is considered to be the time until the next job arrival. The thresholds are given by the algorithm given in Section 2.

For a given set of processors, we compare the performance of these policies by plotting the mean delay (queueing plus service) of each policy as a function of the arrival rate of the system, denoted by λ. We also plot an unachievable lower bound on the expected response time which is obtained by solving a queue length dependent M/M/1 system where the service rate when there are n jobs is given by $\sum_{i=1}^{\min(n, N)} \mu_i$. We use the exact analysis found in Cooper (1976) for the **FAS** policy and use simulation for each of the other policies. For each simulation point we also show ninety percent confidence intervals.

3.1 Performance Results

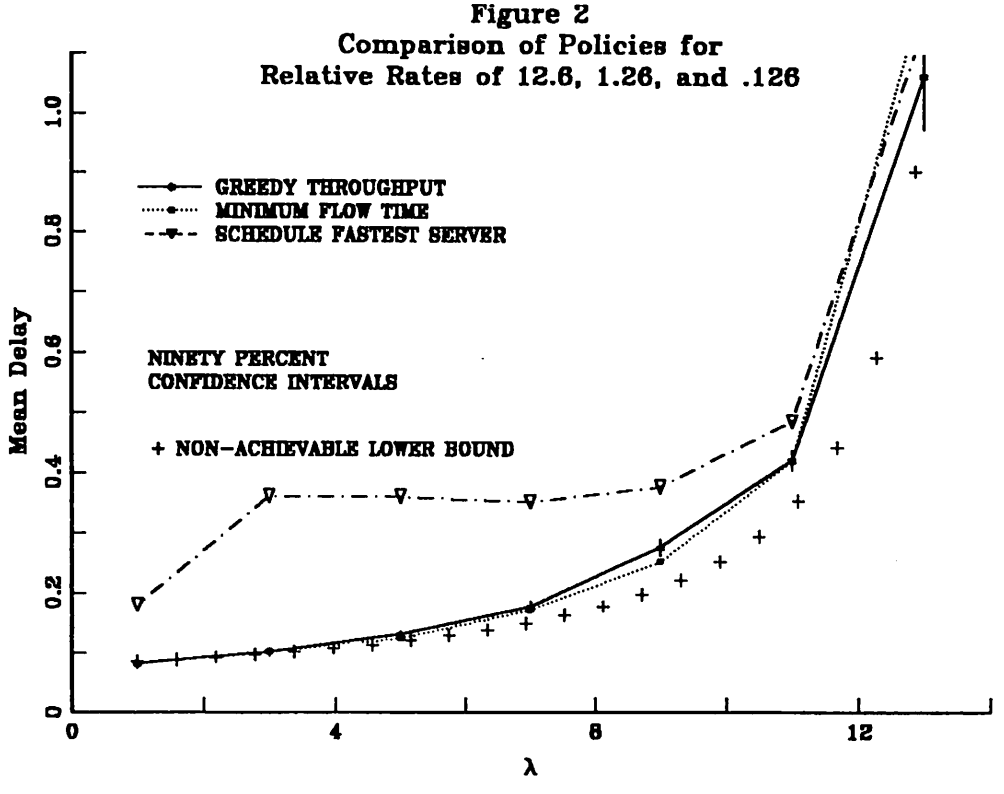
In this section we compare the delay performance of the three policies described in the previous section. We concentrate on systems where the relative rates of the servers have a wide spread.



In Figure 1 we have shown the delays that are obtained when there are three servers with geometrically decreasing rates of 8, 4, and 2. Shown in the Figure are the mean delays for each of the policies obtained from simulation with ninety percent confidence intervals. In looking at the graph one sees that for these relative rates there is not a substantial difference between any of the three policies. This results from the fact that the thresholds for the threshold policies are not large. In particular for **MFT** the thresholds are are 1,2 and 5 for all input rates and are all equal to 1 for **FAS**. The threshold values for various input rates for **GT** are given in Table 1.

λ	*Thresholds*
1	1,1,4
3	1,1,2
5	1,1,2
7	1,1,2
9	1,1,2
11	1,1,2
13	1,1,2

Table 1
Thresholds for Relative Rates of 8, 4, 2



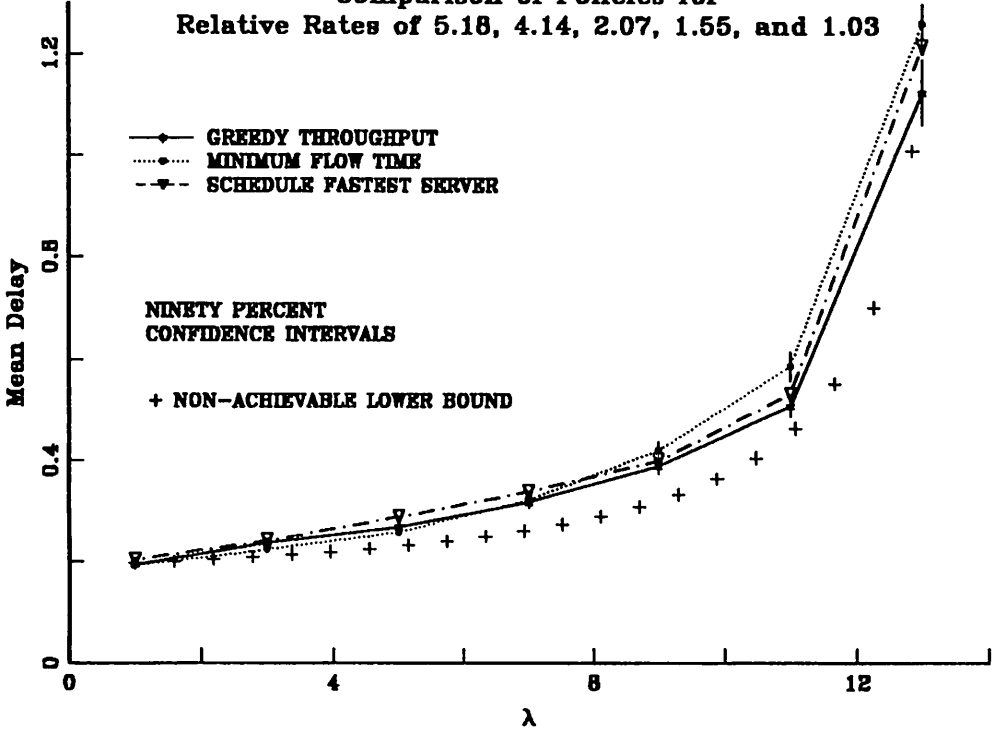
In Figure 2 the spread of the service speeds are again geometric but this time by a factor of 10 instead of 4. The sum of the service rates of the three servers remains 14. Threshold values for the MFT policy are 1, 10 and 108.

λ	*Thresholds*
1	1,7,30
3	1,5,15
5	1,4,11
7	1,4,9
9	1,3,8
11	1,3,7
13	1,1,2

Table 2
Thresholds for Relative Rates of 12.6, 1.26, .126

In this case we find a startling difference in FAS from that of the other policies. Initially one might think that the very large delay value for the arrival rate of 3.5 was in error but this rather anomalous behavior can be readily explained by the fact that FAS schedules even the very slow server (the one with a rate of .126) in conditions where the mean delay of the system would be decreased if customers waited until the fastest server became available.

Figure 3
Comparison of Policies for
Relative Rates of 5.18, 4.14, 2.07, 1.55, and 1.03



In Figure 3 we have increased the number of servers to five and have chosen relative rates of 100, 80, 40, 30 and 20 normalized to sum to 14 so as to be comparable to the previous Figures. We show the thresholds for the **GT** policy in Table 3. The thresholds for the **MFT** policy are 1,2,4,6 and 10 respectively.

λ	*Thresholds*
1	1,1,2,2,4
3	1,1,2,2,3
5	1,1,2,2,3
7	1,1,2,2,3
9	1,1,1,2,2
11	1,1,1,2,2
13	1,1,1,2,2

Table 3
Thresholds for Relative Rates of 5.18, 4.14, 2.07, 1.55, 1.03

The **GT** policy exhibits better performance than **MFT**. This is due to the fact **MFT** is not greedy enough.

4. Future Research

There are extensions of the model presented in this work that are of interest. One extension would concern the scheduling policy that maximized the number of job completions before a deadline for a system in which the servers differed not only in speed but also in their capacities. This would more realistically model a disaster evacuation in which the transport vehicles had different loading capacities. Other extensions concern the relaxation of the exponential distribution assumptions for both servicing rates and for the deadline. As the memoryless property of the exponential distribution was necessary in our derivation of the optimal policy, this extension appears to be mathematically difficult. In a computer system it is often possible to preempt jobs from service. Associated with preemption, however, is a non-zero cost (the time necessary to move preempted jobs) that results from the overheads of process switching. Extending our model to include preemptions with this cost is an interesting problem.

References

Agrawala, A., E. Coffman, M. Garey and S. Tripathi. 1984. A Stochastic Optimization Algorithm Minimizing Expected Flow Times on Uniform Processors. *IEEE Transactions on Computers*, Vol. C-33, No. 4, pp. 351-356.

Chow, Y. and W. Kohler. 1979. Models for Dynamic Load Balancing in a Heterogeneous Multiple Processor System. *IEEE Transactions on Computers*, Vol. C-28, No. 5, pp. 354-361.

Cooper, R. 1976. Queues with Ordered Servers that Work at Different Rates. *Opsearch,* Vol. 13, No. 2, pp. 69-77.

Kumar, P. R. and Walrand, J. 1985. Individually Optimal Routing in Parallel Systems. *J. Appl. Prob.*, Vol 22, pp. 989-995.

Larsen, R. and A. Agrawala. 1983. Control of a Heterogeneous Two-Server Exponential Queueing System. *IEEE Transactions on Software Engineering*, Vol. SE-9, No. 4, pp. 521-527.

Lin, W. and P. Kumar. 1984. Optimal Control of a Queueing System with Two Heterogeneous Servers. *IEEE Transactions on Automatic Control* , Vol. AC-29, No. 8, pp. 696-703.

Nelson, R. and Towsley, D. 1984. Approximating the Mean Delay of a Multiple Server Queue using Threshold Scheduling. IBM Research Report RC 10912. To be published in Operations Research.

Rubinovitch, M. 1985. The Slow Server Problem: A Queue with Stalling. *J. Appl. Prob.* Vol 22, pp. 879-892.

Walrand, J. 1984. A Note on Optimal Control of a Queueing System with two Heterogeneous servers. *Systems and Control Letters* Vol 4, pp. 131-134.

Weber, R. 1982. Scheduling Jobs with Stochastic Processing Requirements on Parallel Machines to Minimize Makespan or Flow Time. *J. Appl. Prob.*, Vol. 19, pp. 167-182.

Weiss, G. and M. Pinedo. 1980. Scheduling Tasks with Exponential Service Times on Non-identical Processors to Minimize Various Cost Functions. *J. Appl. Prob.*, Vol. 17, pp. 187-202.