Optimal Time-Varying Load Sharing for Divisible Loads

JEEHO SOHN
THOMAS G. ROBERTAZZI, Senior Member, IEEE
University of Stony Brook

A load sharing problem involving the optimal load allocation of divisible loads in a distributed computing system consisting of N processors interconnected through a bus-oriented network is investigated. For a *divisible load*, the workload is infinitely divisible so that each fraction of the workload can be distributed and independently computed on each processor. For the first time in divisible load theory, an analysis is provided in the case when the processor speed and the channel speed are time varying due to background jobs submitted to the distributed system with nonnegligible communication delays. A numerical method to calculate the average of the time-varying processor speed and the channel speed and an algorithm to find the optimal allocation of the workload to minimize the total processing finish time are proposed via a deterministic analysis. A stochastic analysis which makes use of Markovian queueing theory is introduced for the case when arrival and departure times of the background jobs are not known.

Manuscript received November 19, 1996; revised September 29, 1997.

IEEE Log No. T-AES/34/3/06023.

This research was supported in part by the BMDO/IST under the U.S. Office of Naval Research under Grant N00014-91-J-4063.

Authors' current addresses: J. Sohn, AT&T, 200 Laurel Ave., Middletown, NJ 07748; T. G. Robertazzi, Dept. of Electrical Engineering, University of Stony Brook, Stony Brook, NY 11794.

I. INTRODUCTION

Divisible loads are computing loads that can be arbitrarily partitioned among a number of (possibly physically distributed) processors so as to gain the benefits of parallel processing to minimize solution time. It is particularly well suited for a large family of problems involving data parallelism in signal, image, and data processing. Specific aerospace applications include sensor networks, Kalman filtering and matched filtering. There has previously been a large amount of work on loads that are not divisible [10, 20]. Since 1988 there has been an increasing amount of theory on divisible loads [19].

Two aspects of divisible load theory, as it has been developed to date, are particularly noteworthy. One noteworthy aspect is its tractability. Simply by working in a continuous time framework and making relatively simple assumptions a linear model results. Just as the basic linear models of electric circuit theory and queueing theory lead to a rich set of results in their respective fields, the linear modeling of divisible loads also produces a broad and elegant theory.

The second noteworthy aspect of divisible load theory is that the model developed to date is a completely deterministic one. No statistical or probabilistic assumptions are made in a typical divisible load analysis. This is significant as probabilistic assumptions can often be the Achilles heel of a performance evaluation of a parallel processing system. Such probabilistic evaluations can suffer from a lack of experimental data to support them or too specific assumptions due to a specific set of experimental measurements. Thus Section V of this work is unusual in that it is the first attempt to integrate divisible load theory and queueing theory.

The study of divisible load theory started from the consideration of intelligent sensor networks by Cheng and Robertazzi [11]. An *intelligent sensor* is a single-processor-based unit which can make measurements, compute, and communicate with other intelligent sensors. The concept of the intelligent sensor network can be extended to the case of a multiple processor environment. The main problem in this research is to determine the optimal schedule of the workload distribution to the processors so as to minimize the problem solution time.

In [11], recursive expressions for calculating the optimal load allocation for linear daisy chains of processors were presented. This is based on the simplifying premise that for an optimal allocation of load, all processors must stop processing at the same time. Intuitively, this is because otherwise some processors would be idle while others were still busy. Analogous solutions have been developed for tree networks [12] and bus networks [2, 3]. Asymptotic solutions for systems with large or even an infinite number of processors and limitations in performance when adding processors appear in [4, 14]. Closed-form solutions were presented in [1] for bus and tree architectures where processor and link speeds are homogeneous. In [16], the concept of an equivalent processor that behaves identically to a collection of processors in the context of a linear daisy chain of processors and a proof that, for a linear daisy chain of processors load sharing a divisible load, the optimal solution involves all processors stopping at the same time are introduced. An analytic proof for bus networks that for a minimal processing time all processors must finish computing at the same time is shown in [18]. Previous proofs were heuristic. In [17], a more sophisticated load-sharing strategy is proposed for bus networks that exploits the special structure of divisible load theory to yield a smaller processing time when a series of jobs are submitted to the network. The equivalence of first distributing load either to the left or to the right from a point in the interior of a linear daisy chain is demonstrated in [13]. Optimal sequences of load distribution in tree networks are described in [6, 15]. Load distribution strategies for tree networks [5], linear daisy chains [7], 2D meshes [8], and hypercubes [9] have appeared.

All the previous works investigated divisible load theory under the assumption that a processor can compute only a single job at a time. Under this assumption, the next job can be served only after the processor finishes the computation of the currently running job. However, most practical time-sharing computer systems can handle more than one job at a time. It is therefore natural that a study of divisible load theory in multiprogrammed and multiprocessor environments is necessary. Another key difference with respect to previous works is that the processor speed and the channel speed will be considered to vary with time while they remain constant in the previous works. The processors, in this paper are assumed to be multiprogrammed so that there are a number of jobs running in the *background* in addition to the divisible load of interest. These background jobs consume processor and link resources so that the divisible load of interest may see time-varying processor and link speed. It is immaterial for the purposes of this work whether the background jobs are divisible or indivisible. The processor speed and the channel speed depend on the number of jobs which are currently served under a processor or transmitted through a channel. When there are a large number of jobs running in a processor, the processor speed for a specific job of interest becomes slower than when it has fewer jobs. The channel speed also becomes slower when there are a large number of background jobs related transmissions passing through a link than when there are fewer transmissions using the links.

The purpose of this work is to determine the optimal fraction of the entire workload to be distributed to each processor to achieve the minimal processing time when the processor speed and the channel speed are time-varying variables. To determine the optimal fraction of the workload deterministically, the processor speed and the channel speed over the duration of the divisible load computation must be known in advance before the load originating processor starts distributing the workload to each processor. If the exact arrival time and departure time of the background jobs are known, one can determine the exact time-varying processor speed and the channel speed. This is suitable for *production jobs* that are performed in a system repeatedly for a known period. If the arrival and the departure times of the background jobs are not known, but the stochastic arrival process and the stochastic departure process of the jobs can be assumed to be Markovian, the optimal fraction of the workload can still be found by a stochastic analysis which makes use of well known Markovian queueing theory. In this work an optimal load sharing algorithm and a numerical method to find the optimal fraction of the entire workload for the minimal processing time is presented by deterministic analysis when the arrival and departure times of the background jobs are known and by stochastic analysis when the arrival and departure times of the background jobs are not known.

This paper is organized as follows. The load-sharing algorithm for the determination of the optimal load allocation for three types of time-varying cases (time-varying processor speed while the channel speed remains constant, time-varying channel speed while the processor speed is constant, and time-varying processor speed and channel speed) are presented in Sections II, III, and IV, respectively. In Section V, a different load-sharing algorithm to find the optimal fraction of the workload via a stochastic analysis when the arrival and departure times of the background jobs are not known is proposed. Performance evaluation results appear in Section VI. Finally, this paper concludes with Section VII.

II. TIME-VARYING PROCESSOR SPEED

The distributed computing system to be considered here consists of a control processor for distributing the workload and N processors attached to a linear bus as in Fig. 1. New arriving measurement data are distributed to each processor under the supervision of the control processor. The control processor distributes the workload among the N processors interconnected through a bus-type communication medium in order to obtain the benefits of parallel processing. Note that the control processor is a network processor which does no processing itself and only distributes the workload. Each processor is a multiprogrammed

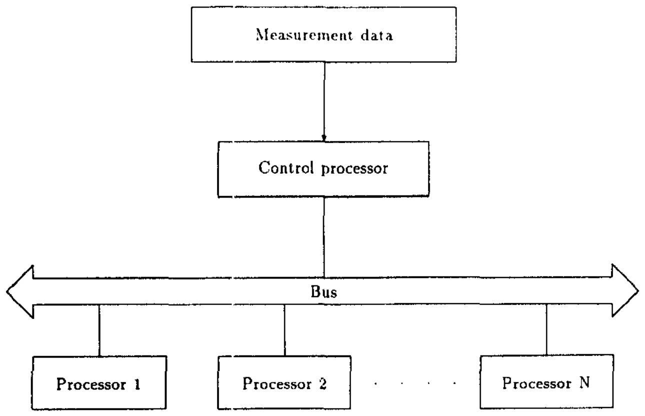


Fig. 1. Bus network with load origination at control processor.

processor that can simultaneously process multiple jobs. Thus the processor speed varies with time and it depends on the amount of workload. The processor speed and the channel speed vary under the following processor sharing rule: the processor (the channel) devotes all its computational power (transmission power) evenly to each job. That is, if there are m jobs running under a certain processor, each job receives $1/m$ of the full computational power of the processor. This behavior is similar to a fair resource scheduling policy as used in UNIX systems. It is assumed here that there is no limitation of the number of jobs to be simultaneously processed in a single processor, even though the processor speed for one particular job will be very slow if there are a large number of jobs running simultaneously under the processor. We assume that background jobs start and terminate simultaneously across all processors and that negligible bus communication is needed to support their running. The main problem in this work is to find the optimal fraction of a divisible load which is distributed to each of N processors to minimize the processing finish time when the communication delay is nonnegligible.

The following notations are used throughout this paper.

α_n	The fraction of the entire processing load that is assigned to the nth processor (P_n).
w_n	Inverse of maximum computing speed of P_n.
$w_n(t)$	Inverse of computing speed of P_n applied to divisible load of interest when computing speed is time varying.
Z	Inverse of maximum channel speed of bus.
$Z(t)$	Inverse of channel speed of bus applied to divisible load of interest when channel speed is time varying.
T_{cp}	Computational load in time, i.e., time it takes for P_n to process entire load when $w_n = 1$.
T_{cm}	Communication load in time, i.e., time it takes to transmit entire set of data over channel when $Z = 1$.
T_n	Time for P_n to complete reception of corresponding fraction of load from distributing processor.
T_f	Processing finish time of entire processing load, assuming load is delivered to origination processor at time zero.

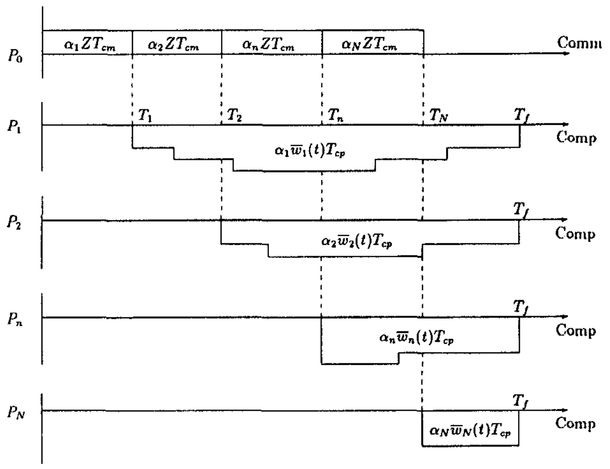


Fig. 2. Timing diagram for bus network with time-varying processor speed.

The timing diagram for the bus network with load origination at a control processor is depicted in Fig. 2. In this timing diagram, communication time appears above the axis and computation time appears below the axis. In this section, the channel speed is assumed to be a constant while the computing speed of each processor is assumed to be time varying. The channel speed is time varying in later sections.

At any time, the processor effort available for the divisible loads of interest varies because of background jobs which consume processor effort. These background jobs can arrive at or terminate on the processors at any time during the computation of the divisible load that the control processor is going to distribute. The arrival and departure times of the background jobs over the interval during which the divisible load is processed, however, should be exactly known. This is the reason that this section and Sections III and IV represent deterministic models of the load-sharing problem. When the arrival and departure times are unknown and the statistics of the arrival and departure process of the jobs are known to be Markovian, then this load-sharing problem can be stochastically analyzed as in Section V.

At time $t = 0$, the originating processor (the control processor in this case) transmits the first fraction of the workload to P_1 in time $\alpha_1 Z T_{cm}$. The control processor then transmits the second fraction of the workload to P_2 in time $\alpha_2 Z T_{cm}$, and so on. After P_1 completes receiving its workload from the control processor (an amount of α_1 of the entire load), P_1 can start computing immediately and it will take a time of $T_f - T_1$ to finish. Here $T_1 = \alpha_1 Z T_{cm}$. The second processor P_2 also completes receiving the workload from the control processor at time $T_2 = (\alpha_1 + \alpha_2)ZT_{cm}$ and it will start computing for a duration of $T_f - T_2$ of time. This procedure continues until the last processor. For optimality, all the processors must finish computing at the same time. Intuitively, this is because otherwise the processing time could be improved by transferring the load from busy processors to idle ones. An analytical proof of this appears in [18].

Now let us represent those intervals of the computation time $T_f - T_1, T_f - T_2, \ldots, T_f - T_N$, carefully. The interval $T_f - T_n$ for P_n to compute the nth fraction of the entire load can be expressed as

$$T_f - T_n = \alpha_n \overline{w}_n(t) T_{cp} \qquad n = 1,2,\ldots,N \tag{1}$$

where $\overline{w}_n(t)$ is defined as the inverse of the time average of the applied computing speed of P_n in the interval (T_n, T_f). Since $w_n(t)$ is defined as *the inverse* of the computing speed, to calculate the time average of $w_n(t)$ one must invert $w_n(t)$ first to make it proportional to the actual computing speed and take the time average, and then invert it again. That is,

$$\overline{w}_n(t) = \left(E\left\{\frac{1}{w_n(t)}\right\}\right)^{-1} = \frac{T_f - T_n}{\int_{T_n}^{T_f} \frac{1}{w_n(t)} dt}. \tag{2}$$

The diagrams for the computing speed of P_n are depicted in Fig. 3(a), (b) and (c). Consider Fig. 3(a), (b), and (c) in reverse order. Fig. 3(c) shows the process which is *proportional* to the computing speed of P_n. When the processor is idle in the interval (t_0, t_1), the load that is delivered from the control processor will receive the full computational power of P_n. Therefore, the computing speed of P_n in the interval (t_0, t_1) for the load from the control processor is $1/w_n$ where w_n is the *inverse* of the maximum computational power of P_n. When there is one background job running in the processor in the interval (t_1, t_2) due to the arrival of one background job at time $t = t_1$, the computational power of P_n is equally divided by two so that each job, one background job, and the divisible load from the control processor, can receive half of the full computational power of P_n. That is, the computing speed of P_n in the interval (t_1, t_2) for each job is $\frac{1}{2}1/w_n$. Likewise, when there are two background jobs running in the processor in the interval (t_2, t_3) due to the additional arrival of a background job at time $t = t_2$, the computational power of P_n is equally divided by three so that each job, two background jobs, and the divisible load from the control processor, can receive one-third of the full computational power of P_n. The computing speed of P_n in the interval (t_2, t_3) for each job is $\frac{1}{3}1/w_n$. When the processor finishes

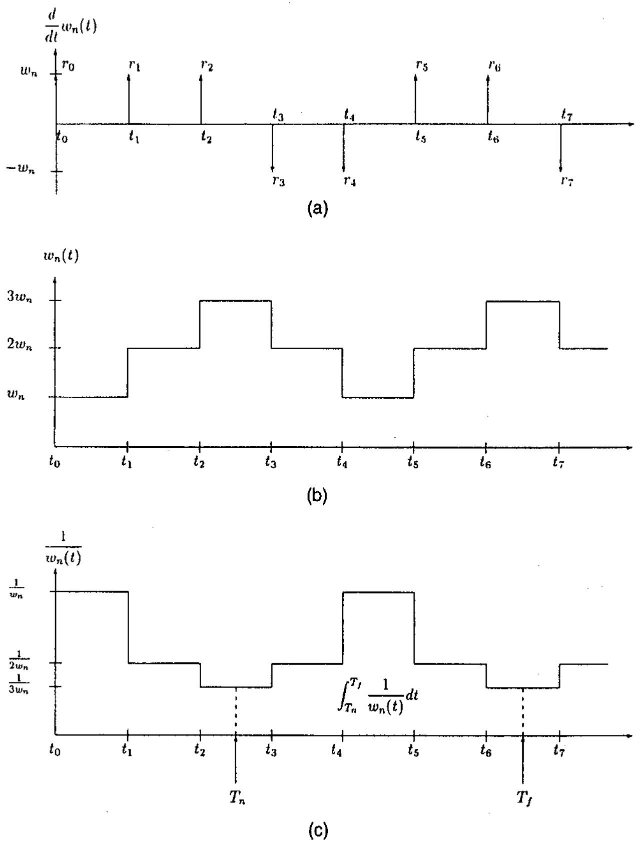


Fig. 3. (a) Derivative of timing process which is inversely proportional to computing speed. (b) Timing process which is inversely proportional to computing speed. (c) Timing process which is proportional to computing speed.

the computation of one of the background jobs at time $t = t_3$, the computing speed of the P_n for each job (at this time, there are two jobs running in the processor, one a background job and the other a divisible load fragment from the control processor) speeds up back to $\frac{1}{2}1/w_n$.

Fig. 3(b) shows the process which is *inversely proportional* to the computing speed of P_n. In other words, Fig. 3(b) is just the inverse of Fig. 3(c). Fig. 3(a) is the derivative of Fig. 3(b). This represents the arrival and departure time of the background jobs. The upright impulses $(r_0, r_1, r_2, r_5, r_6)$ represent the arrival of each background job and the upside down impulses (r_3, r_4, r_7) represent the departure or service completion of each background job. What is deterministic in this section is that the time of each arrival and departure of the background jobs is deterministically known. That is, the time $t_0, t_1, t_2, \ldots$, etc. should be all known at time $t = 0$. This condition can be true of a production system repetitively running the same jobs. The height of each impulse is $+w_n$ for the ones which correspond to the arrivals and $-w_n$ for the ones which corresponds to the departure of the background job. This is because one arrival of a background job causes the computing speed to change from $(1/m)1/w_n$ to $(1/m + 1)1/w_n$ in Fig. 3(c) so the inverse speed changes from mw_n to $(m+1)w_n$ in Fig. 3(b) for any integer m. The same explanation can be applied to the departure of background jobs.

Let us now find the expressions for Fig. 3(a), (b), and (c). The expression for Fig. 3(a) is

$$\frac{d}{dt}w_n(t) = \sum_{k=0}^{\infty} r_k \delta(t-t_k)w_n \tag{3}$$

where

$$r_k = \begin{cases} +1, & \text{for arrival} \\ -1, & \text{for departure} \end{cases}.$$

The following equation represents Fig. 3(b)

$$w_n(t) = \sum_{k=0}^{\infty} r_k u(t-t_k)w_n. \tag{4}$$

Here $u(t)$ is the unit step function. A little thought yields an expression for Fig. 3(c):

$$\frac{1}{w_n(t)} = \sum_{k=0}^{\infty}\left(\sum_{j=0}^{k} r_j\right)^{-1} [u(t-t_k)-u(t-t_{k+1})]\frac{1}{w_n}. \tag{5}$$

The next step is to find the time average of $w_n(t)$ in the interval (T_n,T_f). To find $\overline{w}_n(t)$, it is necessary to find $\int_{T_n}^{T_f} 1/w_n(t)dt$ from (2)

$$\int_{T_n}^{T_f}\frac{1}{w_n(t)}dt = \frac{T_f}{w_n(T_f)} - \frac{T_n}{w_n(T_n)} - \sum_{k=x_n+1}^{x_f}\left(\frac{1}{w_n(t_k)}-\frac{1}{w_n(t_{k-1})}\right)t_k. \tag{6}$$

See Appendix A for details. Therefore,

$$\overline{w}_n(t) = \frac{T_f - T_n}{\frac{T_f}{w_n(T_f)} - \frac{T_n}{w_n(T_n)} - \sum_{k=x_n+1}^{x_f}\left(\frac{1}{w_n(t_k)}-\frac{1}{w_n(t_{k-1})}\right)t_k}. \tag{7}$$

From (1), one can also find the expression for α_n

$$T_f - T_n = \alpha_n \overline{w}_n(t)T_{cp} = \alpha_n T_{cp}\frac{T_f - T_n}{\int_{T_n}^{T_f}\frac{1}{w_n(t)}dt}. \tag{8}$$

Thus,

$$\alpha_n = \frac{1}{T_{cp}}\int_{T_n}^{T_f}\frac{1}{w_n(t)}dt = \frac{1}{T_{cp}}\left[\frac{T_f}{w_n(T_f)} - \frac{T_n}{w_n(T_n)} - \sum_{k=x_n+1}^{x_f}\left(\frac{1}{w_n(t_k)}-\frac{1}{w_n(t_{k-1})}\right)t_k\right]. \tag{9}$$

Here (6), (7), and (9) are functions of T_n and T_f. That is, if T_n and T_f are known, the fraction of the workload for P_n as well as the integral of the applied computing speed of the nth processor and the inverse of the average applied computing speed of P_n in the interval (T_n,T_f) can be found. This problem can be solved by a simple recursive method that can express every α_n as a function of T_f. Let us introduce an algorithm to find the optimal fraction of workload that the control processor must calculate before distributing the load to each processor.

1) Express α_N as a function of T_f from

$$\alpha_N = \frac{1}{T_{cp}}\int_{T_N}^{T_f}\frac{1}{w_N(t)}dt.$$

Since $T_N = (\alpha_1 + \alpha_2 + \cdots + \alpha_N)ZT_{cm} = ZT_{cm}$, T_N is known.

2) Express α_{N-1} as a function of T_f from

$$\alpha_{N-1} = \frac{1}{T_{cp}}\int_{T_{N-1}}^{T_f}\frac{1}{w_{N-1}(t)}dt.$$

Since $T_{N-1} = (1-\alpha_N)ZT_{cm}$, T_{N-1} is a function of α_N and is also a function of T_f.

3) Express α_{N-2} as a function of T_f from

$$\alpha_{N-2} = \frac{1}{T_{cp}}\int_{T_{N-2}}^{T_f}\frac{1}{w_{N-2}(t)}dt.$$

Since $T_{N-2} = (1-\alpha_N-\alpha_{N-1})ZT_{cm}$, T_{N-2} is a function of α_N and α_{N-1}, and is also a function of T_f.

4) This procedure can be continued up to α_1. Then, one can express every α_n as a function of T_f. Finally, by using the normalization equation which states that $\sum_{n=1}^{N}\alpha_n = 1$, all of the α_n, as well as the actual T_f, can be found.

Note that the algorithm, like the ones to follow, starts from time 0 when the initial processor speeds are known as they are a function of past arrivals and departures.

III. TIME-VARYING CHANNEL SPEED

This section considers the opposite situation to that of the previous section. That is, the channel speed is now time varying while the processor speed is constant. This is the case when the channel is shared with other networks. When the channel is idle, the control processor can transmit the measurement data to each processor with the full channel speed. When there is a transmission in the channel from another network, the measurement data transmitted by the control processor will share this channel and it will receive half the speed of the maximum channel capacity. Thus, the channel speed in this section is time varying by the number of transmissions through this channel in a channel (processor-like) sharing manner. Each processor is assumed not to

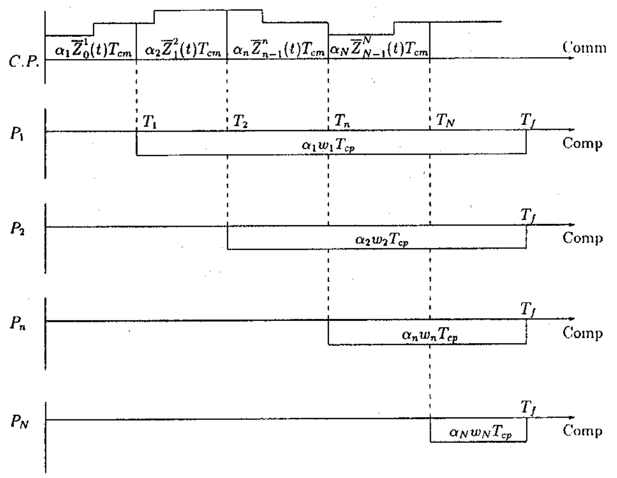


Fig. 4. Timing diagram for bus network with time-varying channel speed.

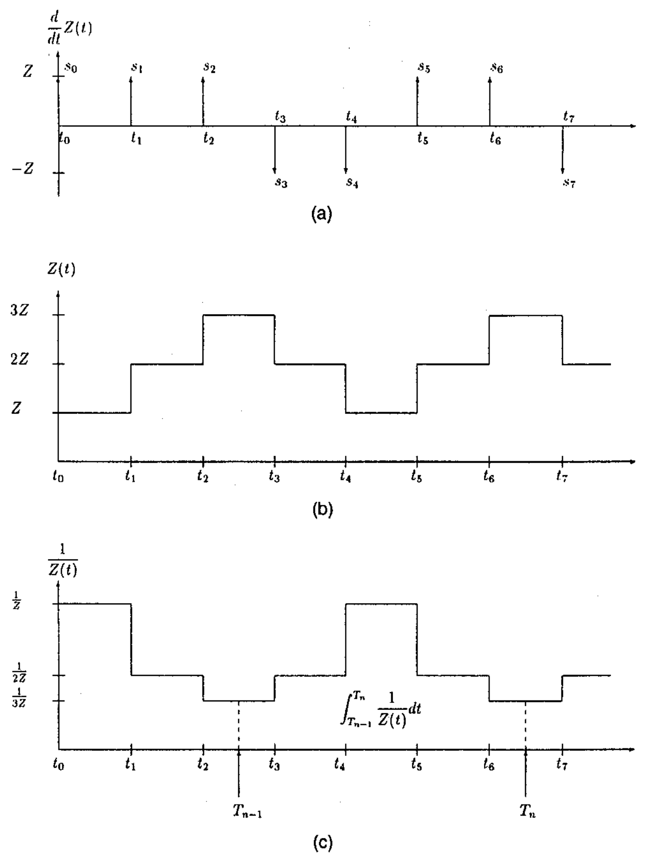


Fig. 5. (a) Derivative of timing process which is inversely proportional to channel speed. (b) Timing process which is inversely proportional to channel speed. (c) Timing process which is proportional to channel speed.

be multiprogrammed. That is, a processor can handle only a single job at a time.

Fig. 4 shows the timing diagram for the case of the time-varying channel speed. At the time origin, the channel may or may not be idle depending on the other networks using the channel. At time $t = 0$, the control processor starts transmitting the first fraction of the workload to P_1 in time T_1. Next the control processor continues to transmit the second fraction of the workload to P_2 and it takes a time $T_2 - T_1$, and so on. Then after P_1 completes receiving its workload from the control processor at time T_1, P_1 can start computing immediately and it will take a time of $T_f - T_1$ to finish. The second processor P_2 also completes receiving the workload from the control processor at time T_2 and it will start computing for a duration of $T_f - T_2$ of time. This procedure continues until the last processor. Again, all the processors must finish computing simultaneously to produce a solution in an optimal amount of time.

The expressions for the computing time for each processor which are the intervals $T_f - T_1, T_f - T_2, \ldots, T_f - T_N$, are more tractable than in the previous section since the computing speed of each processor is not time varying now. One has

$$T_f - T_n = \alpha_n w_n T_{cp} \qquad n = 1,2,\ldots,N. \tag{10}$$

On the other hand, the expressions for the transmission time during which the control processor distributes each fraction of workload to each processor is not as simple since the channel speed is time varying. The transmitting time for each processor from the control processor is

$$T_n - T_{n-1} = \alpha_n \overline{Z}_{n-1}^n(t) T_{cm} \qquad n = 1,2,\ldots,N \tag{11}$$

where $\overline{Z}_{n-1}^n(t)$ is defined as the inverse of the time average of the applied channel speed in the interval (T_{n-1}, T_n). Again, since $Z(t)$ is defined as *the inverse* of the channel speed, to calculate the "time average" of $Z(t)$ one must invert $Z(t)$ first, to make it proportional to the actual channel speed and take the time average, and then invert it again. That is,

$$\overline{Z}_{n-1}^n(t) = \left(E\left\{\frac{1}{Z_{n-1}^n(t)}\right\}\right)^{-1} = \frac{T_n - T_{n-1}}{\int_{T_{n-1}}^{T_n}\frac{1}{Z(t)}dt}. \tag{12}$$

The diagrams for the channel speed are depicted in Fig. 5(a), (b), and (c). A similar explanation as in Fig. 3 can be applied to Fig. 5. The expression for Fig. 5(a) is

$$\frac{d}{dt}Z(t) = \sum_{k=0}^{\infty} s_k \delta(t - t_k) Z \tag{13}$$

where

$$s_k = \begin{cases} +1, & \text{for arrival} \\ -1, & \text{for departure} \end{cases}.$$

The following equations represent Fig. 5(b) and (c)

$$Z(t) = \sum_{k=0}^{\infty} s_k u(t - t_k) Z \tag{14}$$

$$\frac{1}{Z(t)} = \sum_{k=0}^{\infty} \left(\sum_{j=0}^{k} s_j \right)^{-1} [u(t - t_k) - u(t - t_{k+1})] \frac{1}{Z}. \tag{15}$$

Then, the area of the channel speed in the interval (T_{n-1}, T_n) is

$$\int_{T_{n-1}}^{T_n} \frac{1}{Z(t)} dt = \frac{T_n}{Z(T_n)} - \frac{T_{n-1}}{Z(T_{n-1})} - \sum_{k=x_{n-1}+1}^{x_n} \left(\frac{1}{Z(t_k)} - \frac{1}{Z(t_{k-1})} \right) t_k. \tag{16}$$

See Appendix B for details. Therefore,

$$\overline{Z}_{n-1}^{n}(t) = \frac{T_n - T_{n-1}}{\frac{T_n}{Z(T_n)} - \frac{T_{n-1}}{Z(T_{n-1})} - \sum_{k=x_{n-1}+1}^{x_n} \left(\frac{1}{Z(t_k)} - \frac{1}{Z(t_{k-1})} \right) t_k}. \tag{17}$$

From (11), one can also find the expression for α_n

$$\begin{aligned} T_n - T_{n-1} &= \alpha_n \overline{Z}_{n-1}^{n}(t) T_{cm} \\ &= \alpha_n T_{cm} \frac{T_n - T_{n-1}}{\int_{T_{n-1}}^{T_n} \frac{1}{Z(t)} dt}. \end{aligned} \tag{18}$$

Thus,

$$\begin{aligned} \alpha_n &= \frac{1}{T_{cm}} \int_{T_{n-1}}^{T_n} \frac{1}{Z(t)} dt \\ &= \frac{1}{T_{cm}} \left[\frac{T_n}{Z(T_n)} - \frac{T_{n-1}}{Z(T_{n-1})} - \sum_{k=x_{n-1}+1}^{x_n} \left(\frac{1}{Z(t_k)} - \frac{1}{Z(t_{k-1})} \right) t_k \right]. \end{aligned} \tag{19}$$

Also,

$$\begin{aligned} \sum_{i=1}^{n} \alpha_i &= \frac{1}{T_{cm}} \left[\int_0^{T_1} + \int_{T_1}^{T_2} + \cdots + \int_{T_{n-1}}^{T_n} \frac{1}{Z(t)} dt \right] \\ &= \frac{1}{T_{cm}} \int_0^{T_n} \frac{1}{Z(t)} dt \\ &= \frac{1}{T_{cm}} \left[\frac{T_n}{Z(T_n)} - \sum_{k=1}^{x_n} \left(\frac{1}{Z(t_k)} - \frac{1}{Z(t_{k-1})} \right) t_k \right]. \end{aligned} \tag{20}$$

Note that (16), (17), and (19) are functions of T_{n-1} and T_n. That is, if T_{n-1} and T_n are known, the fraction of the workload for P_n as well as the integral of the applied communication speed and the inverse of the average applied communication speed of $Z(t)$ in the interval (T_n, T_f) can be found. Similar to the previous section, this problem can be solved by a simple recursive method that can express every α_n as a function of α_N. Let us introduce an algorithm to find the optimal fraction of workload that the control processor must calculate before distributing the load to each processor.

1) Find T_N from

$$\sum_{i=1}^{N} \alpha_i = 1 = \frac{1}{T_{cm}} \int_0^{T_N} \frac{1}{Z(t)} dt.$$

2) Express T_{N-1} as a function of α_N from

$$\sum_{i=1}^{N-1} \alpha_i = 1 - \alpha_N = \frac{1}{T_{cm}} \int_0^{T_{N-1}} \frac{1}{Z(t)} dt.$$

3) Express α_{N-1} as a function of α_N from

$$T_N - T_{N-1} = \alpha_{N-1} w_{N-1} T_{cp} - \alpha_N w_N T_{cp}$$

since T_N was found in step 1 and T_{N-1} is also a function of α_N.

4) Express T_{N-2} as a function of α_N from

$$\sum_{i=1}^{N-2} \alpha_i = 1 - \alpha_N - \alpha_{N-1} = \frac{1}{T_{cm}} \int_0^{T_{N-2}} \frac{1}{Z(t)} dt$$

since α_{N-1} is a function of α_N.

5) Express α_{N-2} as a function of α_N from

$$T_{N-1} - T_{N-2} = \alpha_{N-2} w_{N-2} T_{cp} - \alpha_{N-1} w_{N-1} T_{cp}$$

since T_{N-1}, T_{N-2}, and α_{N-1} are functions of α_N.

6) This procedure can be continued up to α_1. Then, one can express every α_n as a function of α_N. Finally, by using the normalization equation, all of the α_n and T_f can be found.

IV. TIME-VARYING PROCESSOR SPEED AND CHANNEL SPEED

In the two previous sections, the recursive algorithms to find the optimal fraction of workload and the numerical method to calculate the integrals of the computing speed and the channel speed were introduced in the case of time-varying processor speed and in the case of time-varying channel speed. It is natural at this point to ask if both the computing speed and the channel speed can be time varying. Fig. 6 depicts the timing diagram for the bus network with time-varying processor speed and channel speed. In this case each processor is a multiprogrammed processor that can handle more than one job at a time and the channel is shared with other networks. Alternately one may assume that background jobs

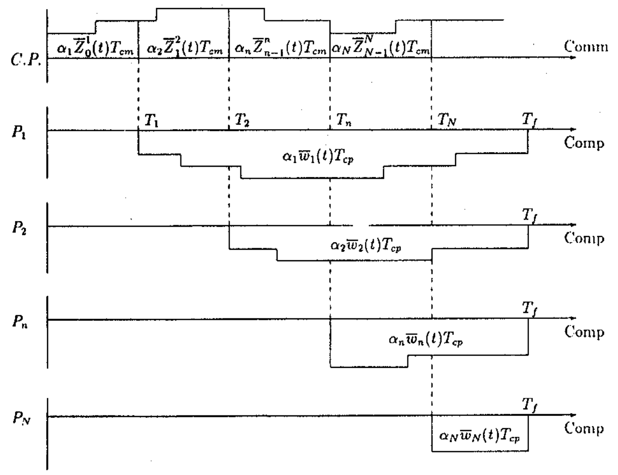

C.P.
$\alpha_1\overline{Z}_0^1(t)T_{cm}$
$\alpha_2\overline{Z}_1^2(t)T_{cm}$
$\alpha_n\overline{Z}_{n-1}^n(t)T_{cm}$
$\alpha_N\overline{Z}_{N-1}^N(t)T_{cm}$
Comm
P_1
T_1
T_2
T_n
T_N
T_f
Comp
$\alpha_1\overline{w}_1(t)T_{cp}$
P_2
T_f
Comp
$\alpha_2\overline{w}_2(t)T_{cp}$
P_n
T_f
Comp
$\alpha_n\overline{w}_n(t)T_{cp}$
P_N
T_f
Comp
$\alpha_N\overline{w}_N(t)T_{cp}$

Fig. 6. Timing diagram for bus network with time-varying processor speed and channel speed.

create communication demands that load the links. To solve the problem in the case of time-varying processor speed and channel speed, the results in the two previous sections are used. Those are

$$\alpha_n = \frac{1}{T_{cp}}\int_{T_n}^{T_f}\frac{1}{w_n(t)}dt$$

$$\sum_{i=1}^{n}\alpha_i = \frac{1}{T_{cm}}\int_0^{T_n}\frac{1}{Z(t)}dt.$$

The numerical methods to calculate the above two equations are the same as in Appendix A and Appendix B. The following is the recursive solution to find the optimal fraction of workload for each processor. It is shown that all the fractions (α_n) can be expressed as a function of α_N.

1) Find T_N from

$$\sum_{i=1}^{N}\alpha_i = 1 = \frac{1}{T_{cm}}\int_0^{T_N}\frac{1}{Z(t)}dt.$$

2) Express T_f as a function of α_N from

$$\alpha_N = \frac{1}{T_{cp}}\int_{T_N}^{T_f}\frac{1}{w_N(t)}dt$$

since T_N was found in step 1.

3) Express T_{N-1} as a function of α_N from

$$\sum_{i=1}^{N-1}\alpha_i = 1-\alpha_N = \frac{1}{T_{cm}}\int_0^{T_{N-1}}\frac{1}{Z(t)}dt.$$

4) Express α_{N-1} as a function of α_N from

$$\alpha_{N-1} = \frac{1}{T_{cp}}\int_{T_{N-1}}^{T_f}\frac{1}{w_{N-1}(t)}dt$$

since T_f and T_{N-1} are also functions of α_N.

5) Express T_{N-2} as a function of α_N from

$$\sum_{i=1}^{N-2}\alpha_i = 1-\alpha_N-\alpha_{N-1} = \frac{1}{T_{cm}}\int_0^{T_{N-2}}\frac{1}{Z(t)}dt$$

since α_{N-1} is also a function of α_N.

6) Express α_{N-2} as a function of α_N from

$$\alpha_{N-2} = \frac{1}{T_{cp}}\int_{T_{N-2}}^{T_f}\frac{1}{w_{N-2}(t)}dt$$

since T_f and T_{N-2} are also functions of α_N.

7) This procedure can be continued up to α_1. Then, one can express every α_n as a function of α_N. Finally, by using the normalization equation, all of the α_n and T_f can be found.

V. STOCHASTIC ANALYSIS OF TIME-VARYING SYSTEMS

It seems that the deterministic analysis of the previous sections are not as realistic as possible because of the constraint that it is applicable only to the case where the exact arrival times and the departure times of the background jobs must be known. It is therefore interesting to pursue a more general analysis that is applicable to practical multiprogrammed and multiprocessor computer systems. A stochastic analysis introduced here will make feasible the determination of the optimal fraction of workload for each processor in more general situations. The exact arrival and departure times of the background jobs submitted to the system in this stochastic analysis are not known. The only necessary knowledge concerning the jobs entering and leaving the network, is the stochastic arrival process and the stochastic departure process in this analysis. If the arrival process is Poisson distributed and the departure process is exponentially distributed, one can adapt well known Markovian queueing theory to this divisible load problem. We assume that job arrival times follow a Poisson process. This is a reasonable first case assumption. The service times are assumed to be either negative exponentially distributed or to follow a general distribution. Thus in the following, two cases of stochastic analysis involving time-varying both the processor speed and the channel speed are presented, one with an $M/M/1$ queueing model and the other one with an $M/G/1$ queueing model.

A. $M/M/1$ Queueing Model

This section starts with the determination of the average computing speed of P_n and the average speed of the shared channel. As in typical queueing models, the arrival rate is defined as λ_{w_n} and λ_Z, and the service rate is defined as μ_{w_n} and μ_Z for P_n and the shared channel, respectively. The service rate $\mu_{w_n}(\mu_Z)$ is proportional to the computing speed of P_n (channel speed) since our server is a linear server. That is, one can write

$$\mu_{w_n} = C_{w_n}\frac{1}{w_n} \tag{21}$$

$$\mu_Z = C_Z\frac{1}{Z} \tag{22}$$

where C_{w_n} and C_Z are constants that are justified in (27) and (28) below. Recall that u_n and Z are defined as the inverse speed of the maximum of P_n and channel, respectively. Let us define $\overline{n}_{w_n}$ and $\overline{n}_Z$ as the *average number of background jobs* in P_n and the average number of transmissions passing through the shared channel, respectively. These are the same as the *average number of customers* in the queueing system with a single queue and is written as

$$\overline{n}_{w_n} = \frac{\rho_{w_n}}{1-\rho_{w_n}} \tag{23}$$

$$\overline{n}_Z = \frac{\rho_Z}{1-\rho_Z} \tag{24}$$

where ρ_{w_n} and ρ_Z are the utilization and are

$$\rho_{w_n} = \frac{\lambda_{w_n}}{\mu_{w_n}} = \frac{\lambda_{w_n}w_n}{C_{w_n}} \tag{25}$$

$$\rho_Z = \frac{\lambda_Z}{\mu_Z} = \frac{\lambda_Z Z}{C_Z}. \tag{26}$$

Note that since $0 \leq \rho_{w_n} < 1$ and $0 \leq \rho_Z < 1$, C_{w_n} and C_Z should be chosen to satisfy the following inequalities

$$0 \leq \lambda_{w_n}w_n < C_{w_n} \tag{27}$$

$$0 \leq \lambda_Z Z < C_Z. \tag{28}$$

Now, one can define the average computing speed of P_n and the average speed of the shared channel as follows

$$\overline{w}_n(t) = (\overline{n}_{w_n}+1)w_n \tag{29}$$

$$\overline{Z}(t) = (\overline{n}_Z+1)Z. \tag{30}$$

One way of explaining these equations is as follows. Suppose that there is no job present in a certain processor at the time when a new divisible load of interest enters the network and is going to be distributed to the processors. Let's consider P_n. Then, P_n can give all its computational power to the divisible load which has just arrived. That is, $\overline{n}_{w_n} = 0$ and $\overline{w}_n$ (at that time) $= (0+1)w_n$. Now, suppose a new background job arrives while the job that was distributed previously is still in progress in P_n. Then, this newly arrived job will receive half of the full computational power in P_n. That is, $\overline{n}_{w_n} = 1$ and $\overline{w}_n$ (at this time) $= (1+1)w_n$. A similar explanation can be applied to the case of the average speed of the shared channel. Therefore, if one substitutes (23) and (25) ((24) and (26)) into (29) ((30)), one can write

$$\overline{w}_n(t) = \frac{C_{w_n}w_n}{C_{w_n}-\lambda_{w_n}w_n} \tag{31}$$

$$\overline{Z}(t) = \frac{C_Z Z}{C_Z-\lambda_Z Z}. \tag{32}$$

Then, the optimal fraction of workload for each processor that minimizes the processing time can be calculated by just replacing the constant computing speed of P_n and the constant channel speed with the above average speed, (31) and (32), into the solution found in [18]. The longer the time interval considered, the more accurate this solution will be.

B. $M/G/1$ Queueing Model

If the computational load of the submitted job is not exponentially distributed and has a general distribution, then the service rate will also be generally distributed. The previous analysis used in the $M/M/1$ queueing model should be modified to that of an $M/G/1$ queueing model. The average number of jobs in P_n and the average number of jobs passing through the shared channel when the computation load is generally distributed can be written as follows from $M/G/1$ queueing theory

$$\overline{n}_{w_n} = \rho_{w_n} + \frac{\rho_{w_n}^2 + \lambda_{w_n}^2\sigma_s^2}{2(1-\rho_{w_n})} \tag{33}$$

$$\overline{n}_Z = \rho_Z + \frac{\rho_Z^2 + \lambda_Z^2\sigma_s^2}{2(1-\rho_Z)} \tag{34}$$

Here, σ_s^2 is the variance of the service time, and ρ_{w_n} and ρ_Z have the same definitions as in the case of $M/M/1$ queueing model. Therefore, the average computing speed of P_n and the average speed of the shared channel are now

$$\overline{w}_n(t) = (\overline{n}_{w_n}+1)w_n = \frac{2C_{w_n}^2 - \lambda_{w_n}^2(w_n^2 - \sigma_s^2C_{w_n}^2)}{2C_{w_n}(C_{w_n}-\lambda_{w_n}w_n)}w_n \tag{35}$$

$$\overline{Z}(t) = (\overline{n}_Z+1)Z = \frac{2C_Z^2 - \lambda_Z^2(Z^2 - \sigma_s^2C_Z^2)}{2C_Z(C_Z-\lambda_Z Z)}Z. \tag{36}$$

Then, the optimal fraction of workload for each processor that minimizes the processing time can be calculated by just replacing the constant computing speed of P_n and the constant channel speed with the above average speed, i.e., (35) and (36), to the solution found in [18]. To do this, let us write the modified solution that the control processor must calculate before distributing the workload to each processor in order to minimize the processing time in the time-varying system

$$k_i(t) = \frac{\overline{w}_i(t)T_{cp}}{\overline{Z}(t)T_{cm} + \overline{w}_{i+1}(t)T_{cp}} \qquad 1 \le i \le N-1 \tag{37}$$

$$\alpha_1 = \left[1 + \sum_{n=1}^{N-1}\left(\prod_{i=1}^{n} k_i(t)\right)\right]^{-1} \tag{38}$$

$$\alpha_n = \prod_{i=1}^{n-1} k_i(t) \cdot \alpha_1 \qquad 2 \le n \le N. \tag{39}$$

Note that the longer the time interval of the divisible load is, the more accurate this substitution will be.

VI. PERFORMANCE EVALUATIONS

Based on the previous results, a number of performance evaluation results were obtained. A simulation was performed in the case where there are three processors connected through the bus ($N = 3$). The simulated run time is from $t = 0$ to $t = 10$. During the 10 units of time, there are 40 randomly generated background arrivals and departures combined. The 10 units of time are sliced into 1000 time slots for the simulation so that each time slot is $\frac{1}{100}$ unit of time. In the following subsections, the simulation results are shown in the cases of time-varying processor speed, time-varying channel speed, time-varying processor and channel speed, and the queueing theory stochastic analysis.

A. Time-Varying Processor Speed

In this subsection, the computing speeds of the three processors are time varying while the channel speed is constant. The computing speeds of the processors are random variables due to randomly generated job arrivals and departures. The channel speed is set to one and the communication load of the divisible job that will be distributed by the control processor is also set to one, and the computational load of the divisible job is set to four ($Z = 1$, $T_{cm} = 1$, $T_{cp} = 4$). Fig. 7 is obtained from the algorithm in Section II. The bottom three curves in Fig. 7 represent α_1, α_2, and α_3, and the most upper curve represents the sum of these αs in the run time $t = 0$ to $t = 10$. The true job finish time occurs when the sum of these αs is equal to one, by the normalization equation, which is in this case between $t = 3.090$ and $t = 3.100$. Table I shows the results of the algorithm in Section II.

These two results are the closest ones obtained from the algorithm in Section II. One can choose either one as a solution and normalize it for implementation. Then, the job computation will be finished no later than $T_f = 3.100$. Alternatively, one can average the two solutions. Note that the true job finish time cannot occur before $T_N = ZT_{cm} = 1$. Thus, there is no data between $t = 0$ and $t = 1$ in Fig. 7. The iterative search procedure should start from T_N and proceeds in the direction of increasing finish time until the sum of the αs equals one.

To check if this simulation and the algorithm is accurate, two methods were used. The first one is an exhaustive grid search in the solution space. Table II shows the results of the exhaustive search. The exhaustive search cannot find the better results even with 500 grid intervals in terms of the job finish time than the one from the algorithm.

The second method is a comparison with true results: Create a set of data such that the processor speed is constant at all times and run the algorithm

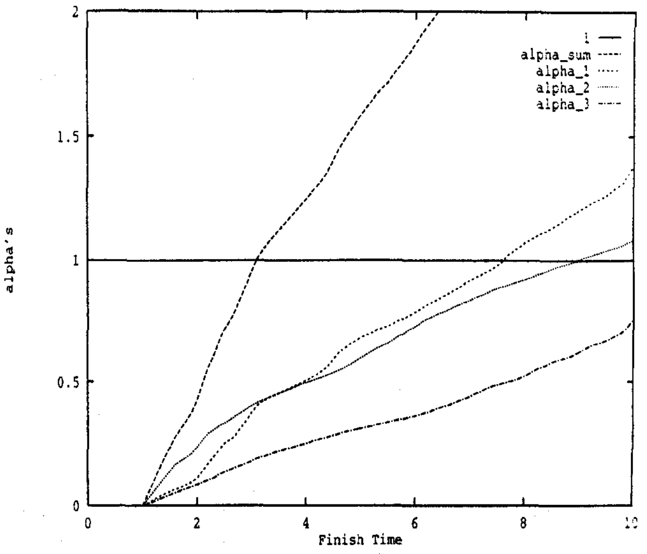


Fig. 7. Job computing finish time for time-varying processor speed.

TABLE I
Result of Algorithm When Processor Speed is Time Varying

T_f	α_1	α_2	α_3	$\sum_{n=1}^{3}\alpha_n$
3.090	0.3992	0.4124	0.1863	0.9979
3.100	0.4023	0.4135	0.1875	1.0033

TABLE II
Result of Exhaustive Search When Processor Speed is Time-Varying

Grid density	T_f	α_1	α_2	α_3
100 × 100 × 100	3.150	0.400	0.410	0.190
200 × 200 × 200	3.130	0.400	0.415	0.185
500 × 500 × 500	3.106	0.402	0.412	0.186

TABLE III
Result of Time-Varying Processor Speed Algorithm with Constant Processor Speed

T_f	α_1	α_2	α_3	$\sum_{n=1}^{3}\alpha_n$
2.04	0.4062	0.3250	0.2600	0.9912
2.05	0.4102	0.3281	0.2625	1.0008

TABLE IV
True Result of Time-Varying Processor Speed System According to Non-Time-Varying System

T_f	α_1	α_2	α_3
2.049180	0.409836	0.327869	0.262295

with this constant processor speed and compare the result with the one from the solution of the non-time-varying system found in [18]. Table III shows the results from the algorithm when all of the processor speed are one ($w_1(t) = w_2(t) = w_3(t) = 1$) in the time-varying system and Table IV shows the true results according to the solution in [18]. All of the true results (T_f, α_1, α_2, and α_3) lie between the two closest results from the algorithm in Table III. If one chooses $T_f = 2.05$ in Table III as a solution of the algorithm, the accuracy is

$$\left(1 - \frac{2.05 - 2.049180}{2.049180}\right) \times 100\% \simeq 99.96\%.$$

B. Time-Varying Channel Speed

The simulation data set in this subsection is the same as in the previous subsection except that the channel speed is now a random variable due to the randomly generated job transmission from other networks and all the processor speeds are constant which are equal to one ($w_1 = w_2 = w_3 = 1$, $T_{cm} = 1$, $T_{cp} = 4$). Fig. 8 is obtained from the algorithm in Section III. Again the bottom three curves represent α_1, α_2, and α_3, and the upper curve represents the sum of αs during the run time $t = 0$ to $t = 6$. The true job finish time occurs when the sum of αs is equal to one, which is between $t = 3.180$ and $t = 3.184$. It cannot occur before T_N which can be calculated from

$$\int_0^{T_N} \frac{1}{Z(t)} dt = T_{cm}$$

and is approximately 2.544 for the given data set here. Table V shows the results from the algorithm in Section III and Table VI shows the results from an exhaustive grid search in the solution space. Again, the results from the algorithm has the smaller job finish time ($T_f = 3.184$) than the one from the exhaustive search ($T_f = 3.188$).

As in the previous subsection, a constant data set was created such that the channel speed was constant at all times ($Z(t) = 1$), and the algorithm was run with this constant channel speed. A comparison was made between these results from the algorithm and the one from [18]. Table VII shows the results

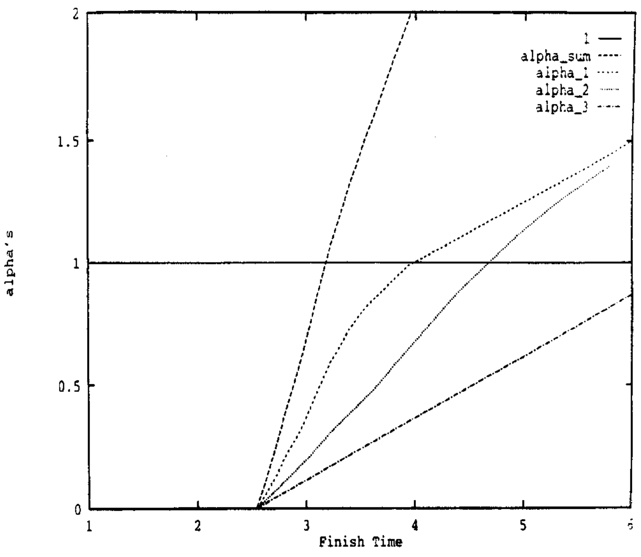


Fig. 8. Job computing finish time for time-varying channel speed.

TABLE V
Result of Algorithm When Channel Speed is Time Varying

T_f	α_1	α_2	α_3	$\sum_{n=1}^{3}\alpha_n$
3.180	0.5500	0.2850	0.1600	0.9950
3.184	0.5535	0.2860	0.1610	1.0005

TABLE VI
Result of Exhaustive Search When Channel Speed is Time Varying

Grid density	T_f	α_1	α_2	α_3
100 × 100 × 100	3.204	0.550	0.290	0.160
200 × 200 × 200	3.195	0.545	0.290	0.165
500 × 500 × 500	3.188	0.556	0.292	0.162

TABLE VII
Result of Time-Varying Channel Speed Algorithm With Constant Channel Speed

T_f	α_1	α_2	α_3	$\sum_{n=1}^{3}\alpha_n$
2.044	0.4085	0.3265	0.2620	0.9970
2.050	0.4125	0.3285	0.2630	1.0040

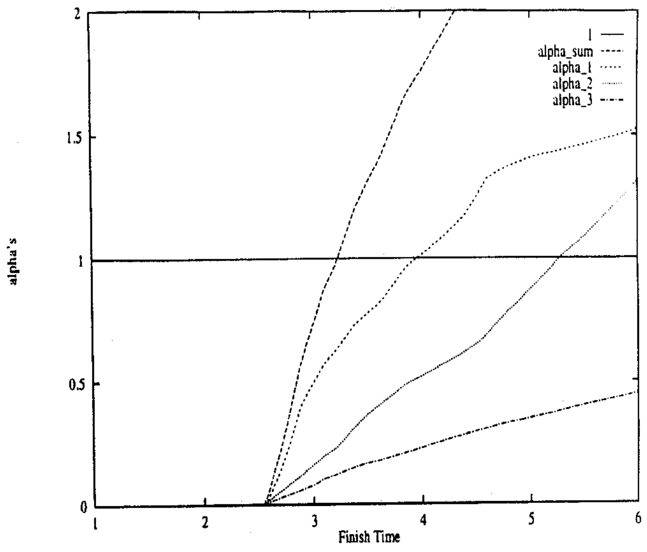


Fig. 9. Job computing finish time for time-varying processor speed and channel speed.

TABLE VIII
True Result of Time-Varying Channel Speed System According to Non-Time-Varying System

T_f	α_1	α_2	α_3
2.049180	0.409836	0.327869	0.262295

TABLE IX
Result of Algorithm When Processor and Channel Speed are Time Varying

T_f	α_1	α_2	α_3	$\sum_{n=1}^{3}\alpha_n$
3.228	0.6368	0.2347	0.1233	0.9948
3.234	0.6408	0.2387	0.1243	1.0038

from the algorithm with constant channel speed and Table VIII shows the true results via [18]. All the true results (T_f, α_1, α_2, and α_3) lie between the two closest results from the algorithm in Table VII. The accuracy is the same as in the previous subsection and is approximately 99.96%.

C. Time-Varying Processor and Channel Speed

This subsection briefly explains the results from a simulation. Both the processor speed and the channel speed are random variables here. It is simulated when $T_{cm} = 1$ and $T_{cp} = 2$. Fig. 9 is obtained from the algorithm in Section IV. Table IX shows the results from the algorithm in Section IV and Table X shows the results from the exhaustive search. The results from the algorithm has the smaller job finish time ($T_f = 3.234$) than the one from the exhaustive search ($T_f = 3.238$).

TABLE X
Result of Exhaustive Search When Processor and Channel Speed are Time Varying

Grid density	T_f	α_1	α_2	α_3
100 × 100 × 100	3.250	0.620	0.250	0.130
200 × 200 × 200	3.245	0.635	0.240	0.125
500 × 500 × 500	3.238	0.640	0.236	0.124

TABLE XI
Result of Time-Varying Processor and Channel Speed Algorithm With Constant Processor and Channel Speed

T_f	α_1	α_2	α_3	$\sum_{n=1}^{3}\alpha_n$
1.416	0.471	0.315	0.210	0.996
1.422	0.477	0.318	0.213	1.008

TABLE XII
True Result of Time-Varying Processor and Channel Speed System According to Non-Time-Varying System

T_f	α_1	α_2	α_3
1.421053	0.473684	0.315789	0.210527

Table XI shows the results from the algorithm with constant processor speed and channel speed ($w_1(t) = w_2(t) = w_3(t) = Z(t) = 1$) and Table XII shows the true results from [18]. All the results (T_f, α_1, α_2, and α_3) lie between the two closest results from the algorithm in Table XI. If one chooses $T_f = 1.422$ in Table XI as a solution of the algorithm, the accuracy is approximately 99.93%.

D. Stochastic Model

Two plots are obtained from the simulation in the stochastic analysis, Figs. 10 and 11. Both the processor speed and the channel speed are time varying and there are 3 processors in the system, and $Z = 1$, $w_1 = 7$, $w_2 = 5$, $w_3 = 3$, $C_z = 10$, and $C_{w_n} = 10$ for all n. The variance of the service time σ_s^2 is equal to zero in the $M/D/1$ queueing model and equal to one in the $M/G/1$ queueing model. In the two plots, the optimal fraction of the workload (αs) and the job finish time (T_f) are drawn against the job arrival rate. The range of the job arrival rate (λ_Z and λ_{w_n}) are from zero to one. In Figs. 10 and 11, the legend is ordered in the order of the curves, i.e., the uppermost curve represents α_3 for $M/M/1$ and the second upper most curve represents α_3 for $M/G/1$ in Fig. 10. It is found that the optimal fractions of the workload (αs) are not sensitive to the choice of queueing models, but they are sensitive with respect to the arrival rates (Fig. 10).

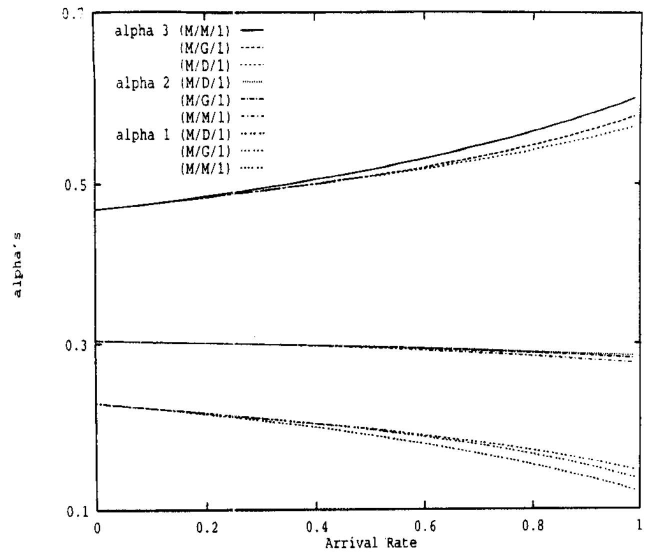


Fig. 10. Optimal fraction of workload (αs) versus arrival rate in stochastic analysis.

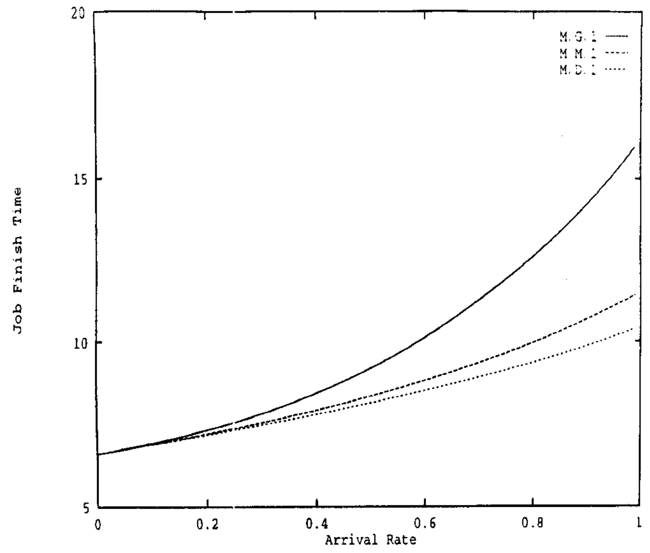


Fig. 11. Job computing finish time in stochastic analysis.

In Fig. 11, it is shown that the job finish time in the $M/M/1$ queueing model takes longer than that in the $M/D/1$ queueing model and shorter than that in the $M/G/1$ queueing model.

VII. CONCLUSIONS

In this paper a numerical method to calculate the average processor speed and the average shared channel speed when these speeds are time varying was found. The algorithm to find the optimal fraction of the workload to minimize the total job computing finish time was also discussed in the deterministic analysis. It was found that the results from the algorithm are accurate. The accuracy was greater than 99.9%.

A simple stochastic analysis using Markovian queueing theory that can handle a more general situation in a time-varying multiprogrammed and multiprocessor environment was introduced here.

Further areas for research would be an extension of the stochastic analysis that can handle more complicated situations, for instance, a job arrival rate that is other than Poisson distributed, an analysis for the service times that might be expressed in terms of the computational load of the job (T_{cp}), and networks other than the bus network, e.g., tree network and hypercube network, etc.

ACKNOWLEDGMENT

The authors wish to thank an anonymous reviewer for the concept of Appendix C.

APPENDIX A

One can calculate the area of the computing speed of the nth processor in the interval (T_n, T_f) as follows:

$$\int_{T_n}^{T_f} \frac{1}{w_n(t)} dt = \int_{T_n}^{T_f} \sum_{k=0}^{\infty} \left(\sum_{j=0}^{k} r_j \right)^{-1} \times [u(t-t_k) - u(t-t_{k+1})] \frac{1}{w_n} dt$$
$$= \frac{1}{w_n} \sum_{k=0}^{\infty} \left(\sum_{j=0}^{k} r_j \right)^{-1} \times \left[\int_{T_n}^{T_f} u(t-t_k)dt - \int_{T_n}^{T_f} u(t-t_{k+1})dt \right]. \tag{40}$$

But,

$$\int_{T_n}^{T_f} u(t-t_k)dt = \begin{cases} T_f - T_n, & \text{if } t_k \le T_n \\ T_f - t_k, & \text{if } T_n < t_k \le T_f \\ 0, & \text{otherwise} \end{cases}$$

$$\int_{T_n}^{T_f} u(t-t_{k+1})dt = \begin{cases} T_f - T_n, & \text{if } t_{k+1} \le T_n \\ T_f - t_{k+1}, & \text{if } T_n < t_{k+1} \le T_f \\ 0, & \text{otherwise} \end{cases}.$$

Let us define new variables x_n and x_f.

DEFINITION 1 The integer x_n is the value of k which satisfies

$$t_k \le T_n < t_{k+1}.$$

DEFINITION 2 The integer x_f is the value of k which satisfies

$$t_k \le T_f < t_{k+1}.$$

For example, in Fig. 3(c), $x_n = 2$ since $t_2 \leq T_n < t_3$ and $x_f = 6$ since $t_6 \leq T_f < t_7$. Then,

$$\int_{T_n}^{T_f} \frac{1}{w_n(t)} dt$$

$$= \frac{1}{w_n}\left[\sum_{k=0}^{x_n}\left(\sum_{j=0}^{k} r_j\right)^{-1}(T_f - T_n) + \sum_{k=x_n+1}^{x_f}\left(\sum_{j=0}^{k} r_j\right)^{-1}(T_f - t_k)\right]$$

$$- \frac{1}{w_n}\left[\sum_{k=0}^{x_n-1}\left(\sum_{j=0}^{k} r_j\right)^{-1}(T_f - T_n) + \sum_{k=x_n}^{x_f-1}\left(\sum_{j=0}^{k} r_j\right)^{-1}(T_f - t_{k+1})\right]$$

$$= \frac{T_f - T_n}{w_n}\left[\sum_{k=0}^{x_n}\left(\sum_{j=0}^{k} r_j\right)^{-1} - \sum_{k=0}^{x_n-1}\left(\sum_{j=0}^{k} r_j\right)^{-1}\right]$$

$$+ \frac{T_f}{w_n}\left[\sum_{k=x_n+1}^{x_f}\left(\sum_{j=0}^{k} r_j\right)^{-1} - \sum_{k=x_n}^{x_f-1}\left(\sum_{j=0}^{k} r_j\right)^{-1}\right]$$

$$- \frac{1}{w_n}\left[\sum_{k=x_n+1}^{x_f}\left(\sum_{j=0}^{k} r_j\right)^{-1} t_k - \sum_{k=x_n}^{x_f-1}\left(\sum_{j=0}^{k} r_j\right)^{-1} t_{k+1}\right]$$

$$= \frac{T_f - T_n}{w_n \sum_{j=0}^{x_n} r_j} + \frac{T_f}{w_n}\left[\left(\sum_{j=0}^{x_f} r_j\right)^{-1} - \left(\sum_{j=0}^{x_n} r_j\right)^{-1}\right]$$

$$- \frac{t_k}{w_n}\sum_{k=x_n+1}^{x_f}\left[\left(\sum_{j=0}^{k} r_j\right)^{-1} - \left(\sum_{j=0}^{k-1} r_j\right)^{-1}\right]$$

$$= \frac{T_f}{w_n \sum_{j=0}^{x_f} r_j} - \frac{T_n}{w_n \sum_{j=0}^{x_n} r_j} - \frac{t_k}{w_n} \times \sum_{k=x_n+1}^{x_f}\left[\left(\sum_{j=0}^{k} r_j\right)^{-1} - \left(\sum_{j=0}^{k-1} r_j\right)^{-1}\right].$$

Note that $(w_n \sum_{j=0}^{x_f} r_j)^{-1}$ is the computing speed of the nth processor at time $t = T_f$. That is,

$$\frac{1}{w_n \sum_{j=0}^{x_f} r_j} = \left.\frac{1}{w_n(t)}\right|_{t=T_f} = \frac{1}{w_n(T_f)}$$

$$\frac{1}{w_n \sum_{j=0}^{x_n} r_j} = \left.\frac{1}{w_n(t)}\right|_{t=T_n} = \frac{1}{w_n(T_n)}$$

$$\frac{1}{w_n \sum_{j=0}^{k} r_j} = \left.\frac{1}{w_n(t)}\right|_{t=t_k} = \frac{1}{w_n(t_k)}$$

$$\frac{1}{w_n \sum_{j=0}^{k-1} r_j} = \left.\frac{1}{w_n(t)}\right|_{t=t_{k-1}} = \frac{1}{w_n(t_{k-1})}.$$

Therefore,

$$\int_{T_n}^{T_f} \frac{1}{w_n(t)} dt = \frac{T_f}{w_n(T_f)} - \frac{T_n}{w_n(T_n)} - \sum_{k=x_n+1}^{x_f}\left(\frac{1}{w_n(t_k)} - \frac{1}{w_n(t_{k-1})}\right) t_k. \tag{41}$$

APPENDIX B

One can calculate the area of the channel speed in the interval (T_{n-1}, T_n) as follows:

$$\int_{T_{n-1}}^{T_n} \frac{1}{Z(t)} dt = \int_{T_{n-1}}^{T_n} \sum_{k=0}^{\infty}\left(\sum_{j=0}^{k} s_j\right)^{-1} \times [u(t - t_k) - u(t - t_{k+1})]\frac{1}{Z} dt$$

$$= \frac{1}{Z}\sum_{k=0}^{\infty}\left(\sum_{j=0}^{k} s_j\right)^{-1} \times \left[\int_{T_{n-1}}^{T_n} u(t - t_k) dt - \int_{T_{n-1}}^{T_n} u(t - t_{k+1}) dt\right]. \tag{42}$$

But,

$$\int_{T_{n-1}}^{T_n} u(t - t_k) dt = \begin{cases} T_n - T_{n-1}, & \text{if } t_k \leq T_{n-1} \\ T_n - t_k, & \text{if } T_{n-1} < t_k \leq T_n \\ 0, & \text{otherwise} \end{cases}$$

$$\int_{T_{n-1}}^{T_n} u(t - t_{k+1}) dt = \begin{cases} T_n - T_{n-1}, & \text{if } t_{k+1} \leq T_{n-1} \\ T_n - t_{k+1}, & \text{if } T_{n-1} < t_{k+1} \leq T_n \, . \\ 0, & \text{otherwise} \end{cases}$$

Then,

$$\int_{T_{n-1}}^{T_n} \frac{1}{Z(t)}dt$$

$$= \frac{1}{Z}\left[\sum_{k=0}^{x_{n-1}}\left(\sum_{j=0}^{k} s_j\right)^{-1}(T_n - T_{n-1}) + \sum_{k=x_{n-1}+1}^{x_n}\left(\sum_{j=0}^{k} s_j\right)^{-1}(T_n - t_k)\right]$$

$$- \frac{1}{Z}\left[\sum_{k=0}^{x_{n-1}-1}\left(\sum_{j=0}^{k} s_j\right)^{-1}(T_n - T_{n-1}) + \sum_{k=x_{n-1}}^{x_n-1}\left(\sum_{j=0}^{k} s_j\right)^{-1}(T_n - t_{k+1})\right]$$

$$= \frac{T_n - T_{n-1}}{Z}\left[\sum_{k=0}^{x_{n-1}}\left(\sum_{j=0}^{k} s_j\right)^{-1} - \sum_{k=0}^{x_{n-1}-1}\left(\sum_{j=0}^{k} s_j\right)^{-1}\right]$$

$$+ \frac{T_n}{Z}\left[\sum_{k=x_{n-1}+1}^{x_n}\left(\sum_{j=0}^{k} s_j\right)^{-1} - \sum_{k=x_{n-1}}^{x_n-1}\left(\sum_{j=0}^{k} s_j\right)^{-1}\right]$$

$$- \frac{1}{Z}\left[\sum_{k=x_{n-1}+1}^{x_n}\left(\sum_{j=0}^{k} s_j\right)^{-1} t_k - \sum_{k=x_{n-1}}^{x_n-1}\left(\sum_{j=0}^{k} s_j\right)^{-1} t_{k+1}\right]$$

$$= \frac{T_n - T_{n-1}}{Z\sum_{j=0}^{x_{n-1}} s_j} + \frac{T_n}{Z}\left[\left(\sum_{j=0}^{x_n} s_j\right)^{-1} - \left(\sum_{j=0}^{x_{n-1}} s_j\right)^{-1}\right]$$

$$- \frac{t_k}{Z}\sum_{k=x_{n-1}+1}^{x_n}\left[\left(\sum_{j=0}^{k} s_j\right)^{-1} - \left(\sum_{j=0}^{k-1} s_j\right)^{-1}\right]$$

$$= \frac{T_n}{Z\sum_{j=0}^{x_n} s_j} - \frac{T_{n-1}}{Z\sum_{j=0}^{x_{n-1}} s_j} - \frac{t_k}{Z} \times \sum_{k=x_{n-1}+1}^{x_n}\left[\left(\sum_{j=0}^{k} s_j\right)^{-1} - \left(\sum_{j=0}^{k-1} s_j\right)^{-1}\right].$$

Note that $(Z\sum_{j=0}^{x_n} s_j)^{-1}$ is the channel speed at time $t = T_n$. That is,

$$\frac{1}{Z\sum_{j=0}^{x_n} s_j} = \left.\frac{1}{Z(t)}\right|_{t=T_n} = \frac{1}{Z(T_n)}$$

$$\frac{1}{Z\sum_{j=0}^{x_{n-1}} s_j} = \left.\frac{1}{Z(t)}\right|_{t=T_{n-1}} = \frac{1}{Z(T_{n-1})}$$

$$\frac{1}{Z\sum_{j=0}^{k} s_j} = \left.\frac{1}{Z(t)}\right|_{t=t_k} = \frac{1}{Z(t_k)}$$

$$\frac{1}{Z\sum_{j=0}^{k-1} s_j} = \left.\frac{1}{Z(t)}\right|_{t=t_{k-1}} = \frac{1}{Z(t_{k-1})}.$$

Therefore,

$$\int_{T_{n-1}}^{T_n} \frac{1}{Z(t)}dt = \frac{T_n}{Z(T_n)} - \frac{T_{n-1}}{Z(T_{n-1})} - \sum_{k=x_{n-1}+1}^{x_n}\left(\frac{1}{Z(t_k)} - \frac{1}{Z(t_{k-1})}\right)t_k. \tag{43}$$

APPENDIX C

Somewhat different parametrizations of the stochastic models of this work eliminate the need for C_{w_n} and C_Z. For instance consider a processor P_i with Poisson arrival rate for *background jobs* of λ_i and a (negative exponential) service rate of μ_i. Then, from basic $M/M/1$ queueing theory, the average number of background jobs, with no divisible job present, is

$$\overline{K}_i = \frac{\lambda_i}{\mu_i - \lambda_i}.$$

Under the fair resource scheduling policy if there is a single divisible job present the service rate applied to the background job is

$$\mu_i \rightarrow \frac{\overline{K}_i}{\overline{K}_i + 1}\mu_i,$$

where μ_i is now the overall service rate.

Thus:

$$\overline{K}_i = \frac{\lambda_i}{\frac{\overline{K}_i}{\overline{K}_i + 1}\mu_i - \lambda_i}.$$

Solving for $\overline{K}_i$, thru a quadratic equation, yields

$$\overline{K}_i = \frac{\lambda_i + \sqrt{\mu_i\lambda_i}}{\mu_i - \lambda_i}.$$

Then, since there are $\overline{K}_i$ background jobs and one divisible job:

$$\overline{w}_i = (\overline{K}_i + 1)w_i = \left(\frac{\mu_i + \sqrt{\mu_i\lambda_i}}{\mu_i - \lambda_i}\right)w_i.$$

Note that the need for C_{w_n} and C_Z has been eliminated.

REFERENCES

[1] Bataineh, S., Hsiung, T., and Robertazzi, T.G. (1994) Closed form solutions for bus and tree networks of processors load sharing a divisible job. *IEEE Transaction on Computers*, **43**, 10 (Oct. 1994), 1184–1196.

[2] Bataineh, S., and Robertazzi, T. G. (1991) Bus oriented load sharing for a network of sensor driven processors. *IEEE Transactions on Systems, Man and Cybernetics*, **21**, 5 (Sept. 1991), 1202–1205.

[3] Bataineh, S., and Robertazzi, T. G. (1991)
Distributed computation for a bus network with communication delays.
In *Proceedings of the 1991 Conference on Information Sciences and Systems*, The Johns Hopkins University, Baltimore, MD, Mar. 1991, 709–714.

[4] Bataineh, S., and Robertazzi, T. G. (1992)
Ultimate performance limits for networks of load sharing processors.
In *Proceedings of the 1992 Conference on Information Science and Systems*, Princeton University, Princeton, NJ, Mar. 1992, 794–799; also in *IEEE Transactions on Aerospace and Electronic Systems* (Oct. 1997).

[5] Bharadwaj, V., Ghose, D., and Mani, V. (1995)
Multi-installment load distribution in tree networks with delays.
IEEE Transactions on Aerospace and Electronic Systems, **31**, 2 (Apr. 1995), 555–567.

[6] Bharadwaj, V., Ghose, D., and Mani, V. (1994)
Optimal sequencing and arrangement in distributed single-level tree networks with communication delays.
IEEE Transactions on Parallel and Distributed Systems, **5**, 9 (Sept. 1994), 968–976.

[7] Bharadwaj, V., Ghose, D., and Mani, V. (1995)
An efficient load distribution strategy for a distributed linear network of processors with communication delays.
Computer and Mathematics with Applications, **29**, 9 (May 1995), 95–112.

[8] Blazewicz, J., and Drozdowski, M. (1996)
The performance limits of a two-dimensional network of load sharing processors.
Foundations of Computing and Decision Sciences, **21**, 1 (1996), 3–15.

[9] Blazewicz, J., and Drozdowski, M. (1996)
Scheduling divisible jobs on hypercubes.
Parallel Computing, **21**, (1996), 1945–1956.

[10] Bokhari, S. H. (1987)
Assignment Problems in Parallel and Distributed Computing.
Boston, MA: Kluwer Academic, 1987.

[11] Cheng, Y. C., and Robertazzi, T. G. (1988)
Distributed computation with communication delays.
IEEE Transactions on Aerospace and Electronic Systems, **24**, 6 (Nov. 1988), 700–712.

[12] Cheng, Y.-C., and Robertazzi, T. G. (1990)
Distributed computation for a tree network with communication delays.
IEEE Transactions on Aerospace and Electronic Systems, **26**, 3 (May 1990), 511–516.

[13] Ghose, D., and Mani, V. (1994)
Distributed computation in a linear network: Closed-form solutions and computational techniques.
IEEE Transactions on Aerospace and Electronic Systems, **30**, 2 (Apr. 1994), 471–483.

[14] Ghose, D., and Mani, V. (1994)
Distributed computation with communication delays: Asymptotic performance analysis.
Journal of Parallel and Distributed Computing, **23** (Nov. 1994), 293–305.

[15] Kim, H. J., Jee, G.-I., and Lee, J. G. (1996)
Optimal load distribution for tree network processors.
IEEE Transactions on Aerospace and Electronic Systems, **32** 2 (Apr. 1996), 607–612.

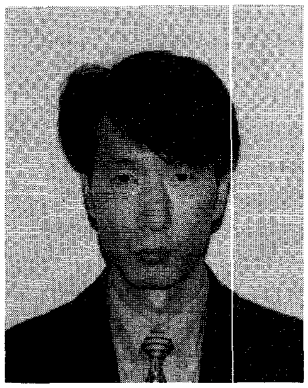
[16] Robertazzi, T. G. (1993)
Processor equivalence for a linear daisy chain of load sharing processors.
IEEE Transactions on Aerospace and Electronic Systems, **29**, 4 (Oct. 1993), 1216–1221.

[17] Sohn, J., and Robertazzi, T. G. (1994)
A multi-job load sharing strategy for divisible jobs on bus networks.
Technical report 697, SUNY at Stony Brook College of Engineering and Applied Science, New York, Aug. 1994.

[18] Sohn, J., and Robertazzi, T. G. (1996)
Optimal divisible job load sharing for bus networks.
IEEE Transactions on Aerospace and Electronic Systems, **32**, 1 (Jan. 1996), 34–40.

[19] Bharadwaj, V., Ghose, D., Mani, V., and Robertazzi, T. G. (1996)
Scheduling Divisible Loads in Parallel and Distributed Systems.
Los Alamitos, CA: IEEE Computer Society Press, 1996.

[20] Shirazi, B. A., Hurson, A. R., and Kavi, K. M. (Eds.) (1995)
Scheduling and Load Balancing in Parallel and Distributed Systems.
Los Alamitos, CA: IEEE Computer Society Press, 1995.



Jeeho Sohn received the B.S. degree in electrical engineering from Yonsei University, Seoul, Korea, in 1987 and the M.S. degree from the University of Colorado, Boulder, in 1989. He received the Ph.D degree in electrical engineering from the University at Stony Brook, Stony Brook, NY, in 1995.

He is presently a Senior Technical Staff Member at AT&T. His research interests include parallel and distributed computing, stochastic and queueing processes, and the performance evaluation of communication and computer systems.

Thomas G. Robertazzi (S'75—M'77—SM'91) received the Ph.D from Princeton University, Princeton, NJ, in 1981 and the B.E.E. from the Cooper Union, New York, NY, in 1977.

He is presently an Associate Professor of Electrical Engineering at the University at Stony Brook. During 1982–1983 he was an Assistant Professor in the Electrical Engineering Department of Manhattan College, Riverdale, NY.

Dr. Robertazzi has served as editor for books for the IEEE Communications Society and an associate editor of the journal *Wireless Networks*. He has authored one book, co-authored a second, and edited a third book in the area of performance evaluation. His research interests involve the performance evaluation of computer and communication systems.