UNIVERSITY AT STONY BROOK

CEAS Technical Report 706

An Optimum Load Sharing Strategy for Divisible Jobs with Time-Varying Processor Speed and Channel Speed

J. Sohn and T.G. Robertazzi

This work was supported by the BMDO/IST under the U.S. Office of Naval Research under grant. no. N00014-91-J4063.

Jan. 11, 1995

An Optimum Load Sharing Strategy for Divisible Jobs with Time-Varying Processor Speed and Channel Speed

Jeeho Sohn and Thomas G. Robertazzi, *Senior Member, IEEE*

Dept. of Electrical Engineering,
University at Stony Brook,
Stony Brook, N.Y. 11794
Tel: (516) 632-8412/8400
Fax: (516) 632-8494

Abstract

In this paper, a load sharing problem involving the optimal load allocation of divisible jobs in a distributed computing system consisting of N processors interconnected through a bus oriented network is investigated. For a *divisible job* 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 job 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 non-negligible 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 solution 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.

Keywords

Divisible job, load balancing, load sharing, multiprogrammed multiprocessor, optimal load allocation, parallel and distributed computing, time-varying computing system.

I. Introduction

In recent years, there has been of great deal of interest in parallel and distributed computing systems because it is a realistic approach for implementing high performance computer systems. Optimal and very efficient load sharing and allocation is essential for achieving minimal processing times. There are many possible ways to classify the load sharing problem. One of them is the classification by the type of job submitted to the system. This leads to *indivisible job* theory and *divisible job* theory. An *indivisible job* is a job that cannot be divided into more than one fragment so that the job must be processed by one processor. There has been an intensive work on this indivisible job theory by many parallel and distributed system researchers [1 – 13]. Only recently has there been interest in multiprocessor scheduling with jobs that need to be assigned to more than one processor [14 – 16]. A *divisible job* is a job that can be arbitrarily partitioned in a linear fashion and can be distributed to more than one processor to achieve a faster solution time. It is particularly suited to the processing of very long linear data files such as occurs in signal and image processing and Kalman filtering.

The load sharing problem in divisible job theory is not trivial as one must take into account the number of processors, the speed of each processor and communication link, and load origination and the network architecture. One important issue for the load sharing problem is a trade-off between the communication time and the computation time. This problem is less important when the size of data file to be transmitted is very small or the communication link speed is very fast so that the time to transmit a typical job can be negligible. However, one must consider the relationship between the communication time and the computation time to achieve the best performance in the load sharing and scheduling problem when the size of data file to be transmitted is very large or the communication link speed is very slow so that the time to transmit a typical job is not negligible. Even though there has been a great deal of work solely on communication and solely on computation, there has not been that much work which deals with both problems. This paper presents a theory for the optimal divisible job load sharing problem which considers both non-negligible communication time and the computation time together.

The study of divisible job theory started from the consideration of intelligent sensor networks by Cheng and Robertazzi [17]. 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 multiprocessor environment. The main problem in this research is to determine the optimal fraction of the workload to assign to each processors. That is, the decision when a network receives a burst of measurement data of what portion of the entire workload should be kept by the distributing processor and what portion of the entire workload should be distributed to each processor in order to minimize the total processing time becomes an important problem.

In [17], 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 [18] and bus networks [19], [20]. Asymptotic solutions for systems with large or even an infinite number of processors and limitations in performance when adding processors appear in [21], [29]. Closed form solutions were presented in [22] for bus and tree architectures where processor and link speeds are homogeneous. In [23], 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 job, the optimal solution involves all processors stopping at the same time are introduced. An analytic proof for bus networks that for a minimal solution time all processors must finish computing at the same time is shown in [24], [25]. Previous proofs were heuristic. In [26], a more sophisticated load sharing strategy is proposed for bus networks that exploits the special structure of divisible job theory to yield a smaller solution time when 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 [28]. Optimal sequences of load distribution in tree networks are described in [27], [30]. A new load distribution strategy for tree networks [31] and linear daisy chains [32] is also discussed.

All the previous works investigated divisible job 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 job 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 job of interest. These background jobs consume processor and link resources so that the divisible job of interest may see time-varying processor and link speed. It is immaterial for the purposes of this paper whether the background jobs are divisible are indivisible. The processor speed and the channel speed depend on the number of jobs which is 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 job 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 solution 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 job 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 are known 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 paper, an optimal load sharing algorithm and a numerical method to find the optimal fraction of the entire workload for the minimal solution time is presented by deterministic analysis when the background jobs' arrival and departure times are known and by stochastic analysis when the background jobs' arrival and departure times 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 section 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 background jobs' arrival and departure times 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 network model 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 is 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 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 run under a certain processor, each job receives $\frac{1}{m}$ of the full computational power of the processor. 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. The main problem in this paper is to find the optimal fraction of a divisible job workload which is distributed to each of N processors to minimize the total solution finish time when the communication delay is nonnegligible.

The following notations will be used throughout this paper:

α_n:	The fraction of the entire processing load that is assigned to the nth processor.
w_n:	The inverse of the maximum computing speed of the nth processor.
$w_n(t)$:	The inverse of the computing speed of the nth processor seen by the divisible job of interest when the computing speed is time-varying.
Z:	The inverse of the maximum channel speed of the bus.
$Z(t)$:	The inverse of the channel speed of the bus seen by the divisible job of interest when the channel speed is time-varying.
T_{cp}:	The computational load of the workload in time, i.e., the time that it takes for the nth processor to process (compute) the entire load when $w_n = 1$.
T_{cm}:	The communication load of the workload in time, i.e., the time that it takes to transmit the entire set of data over the channel when $Z = 1$.
T_n:	The time for the nth processor to complete the receiving of the corresponding fraction of load from the distributing processor.
T_f:	The finish time of the entire processing load, assuming that the load is delivered to the origination processor at time zero.

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 a time-varying. The channel speed will be time-varying in the later sections.

At any time the processor effort available for the divisible jobs of interest varies because of back ground 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 job that the control processor is going to distribute. The arrival and departure times of the background jobs over interval during which the divisible job is processed, however, should be exactly known. This is the reason that this section and section 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 the first processor in time $\alpha_1 Z T_{cm}$. The control processor then transmits the second fraction of the workload to the second processor in time $\alpha_2 Z T_{cm}$, and so on. After the first processor completes receiving its workload from the control processor (an amount of α_1 of the entire load), the first processor 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 also completes receiving the workload from the control processor at time $T_2 = (\alpha_1 + \alpha_2) Z T_{cm}$ 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 solution time could be improved by transferring the load from busy processors to idle ones. An analytical proof of this appears in [24].

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 the nth processor 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 time average of the computing speed of the nth processor 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,

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

The diagrams for the computing speed of the nth processor are depicted 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 the nth processor. When the processor is idle in the interval (t_0, t_1), the job that is delivered from the control processor will receive the full computational power of the nth processor. Therefore, the computing speed of the nth processor in the interval (t_0, t_1) for the job from the control processor is $\frac{1}{w_n}$ where w_n is the *inverse* of the maximum computational power of the nth processor. 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 the nth processor is equally divided by two so that each job, one background job and the job from the control processor, can receive half of the full computational power of the nth processor. That is, the computing speed of the nth processor in the interval (t_1, t_2) for each job is $\frac{1}{2}\frac{1}{w_n}$. 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 the nth processor is equally divided by three so that each job, two background jobs and the job from the control processor, can receive one third of the full computational power of the nth processor. The computing speed of the nth processor in the interval (t_2, t_3) for each job is $\frac{1}{3}\frac{1}{w_n}$. When the processor finishes the computation of one of the background jobs at time $t = t_3$, the computing speed of the nth processor for each job (at this time, there are two jobs running in the processor, one a background job and the other a divisible job fragment from the control processor) speeds up back to $\frac{1}{2}\frac{1}{w_n}$.

Fig. 3(b) shows the process which is *inversely proportional* to the computing speed of the nth processor. 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 $t = 0$. The height of the each impulse is $+w_n$ for the ones which corresponds 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 $\frac{1}{m}\frac{1}{w_n}$ to $\frac{1}{m+1}\frac{1}{w_n}$ in Fig. 3(c) so the speed changes from mw_n to $(m+1)w_n$ in Fig. 3(b) for any integer m. The same explanation can 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 $\int_{T_n}^{T_f} \frac{1}{w_n(t)} dt$ from Eq.(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 Eq.(1), one can also find the expression for α_n.

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

Thus,

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

Note that Eq.(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 the nth processor as well as the integral and the average of the computing speed of the nth processor 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.

i. 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.

ii. 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 function of α_N and is also function of T_f.

iii. 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 function of α_N and α_{N-1}, and is also function of T_f.

iv. 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.

III. Time-Varying Channel Speed

This section will consider 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 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 the first processor in time T_1. Next the control processor continues to transmit the second fraction of the workload to the second processor and it takes a time $T_2 - T_1$, and so on. Then after the first processor completes receiving its workload from the control processor at time T_1, the first processor can start computing immediately and it will take a time of $T_f - T_1$ to finish. The second processor 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. They are

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

On the other hands, 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

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

where $\overline{Z}^n_{n-1}(t)$ is defined as the time average of the channel speed in the interval (T_n, T_f). 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,

$$\begin{aligned} \overline{Z}^n_{n-1}(t) &= \left(E\left\{ \frac{1}{Z^n_{n-1}(t)} \right\} \right)^{-1} \\ &= \frac{T_n - T_{n-1}}{\int_{T_{n-1}}^{T_n} \frac{1}{Z(t)} dt} \end{aligned} \tag{12}$$

The diagrams for the channel speed are depicted 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).

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

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

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 Eq.(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 Eq.(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 the nth processor as well as the integral and the average of the channel speed 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.

i. Find T_N from

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

ii. Express T_{N-1} as a function of α_N from

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

iii. 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 i and T_{N-1} is also function of α_N.

iv. 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.

v. 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 function of α_N.

vi. 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 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 will be used. Those are

$$\begin{aligned} \alpha_n &= \frac{1}{T_{cp}} \int_{T_n}^{T_f} \frac{1}{w_n(t)} dt \\ \sum_{i=1}^{n} \alpha_i &= \frac{1}{T_{cm}} \int_0^{T_n} \frac{1}{Z(t)} dt \end{aligned}$$

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 will be shown that all the fractions (α_n) can be expressed as a function of α_N.

i. Find T_N from

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

ii. 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 i.

iii. Express T_{N-1} as a function of α_N from

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

iv. 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 function of α_N.

v. 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 function of α_N.

vi. 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 function of α_N.

vii. 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 the 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 that will be 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 job problem. We will assume that jobs arrival times follow a Poisson process. This is a reasonable first case assumption. The service times will be 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 will be presented, one with a M/M/1 queueing model and the other one with a M/G/1 queueing model.

A. M/M/1 Queueing Model

This section starts with the determination of the average computing speed of the nth processor 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 the nth processor and the shared channel, respectively. The service rate μ_{w_n} (μ_Z) is proportional to the computing speed of the nth processor (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 Eq.(27) and (28) below. Recall that w_n and Z are defined as the inverse speed of the maximum of the nth processor and channel, respectively. Let us define $\overline{n}_{w_n}$ and $\overline{n}_Z$ as the average number of background jobs in the nth processor 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 $0 \le \rho_{w_n}, \rho_Z < 1$, C_{w_n} and C_Z should be chosen to satisfy the following inequalities.

$$0 \le \lambda_{w_n}w_n < C_{w_n} \tag{27}$$
$$0 \le \lambda_Z Z < C_Z \tag{28}$$

Now, one can define the average computing speed of the nth processor 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 job of interest enters the network and is going to be distributed to the processors. Let's consider processor n. Then, the processor n can give all its computational power to the divisible job which has just arrived. That is, $\overline{n}_{w_n} = 0$ and $\overline{w}_n(\text{at that time}) = (0+1)w_n$. Now, suppose a new background job arrives while the job that was distributed previously is still in progress in processor n. Then, this newly arrived job will receive half of the full computational power in the processor n. That is, $\overline{n}_{w_n} = 1$ and $\overline{w}_n(\text{at this time}) = (1+1)w_n$. A similar explanation can be applied to the case of the average speed of the shared channel. Therefore, if one substitutes Eq.(23) and Eq.(25) (Eq.(24) and Eq.(26)) into Eq.(29) (Eq.(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 minimize the total solution time can be calculated by just replacing the constant computing speed of the nth processor and the constant channel speed with the above average speed, Eq.(31) and Eq.(32), into the solution found in [24]. 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 a M/G/1 queueing model. The average number of jobs in the nth processor 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 the nth processor and the average speed of the shared channel are now

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

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

Then, the optimal fraction of workload for each processor that minimizes the total solution time can be calculated by just replacing the constant computing speed of the nth processor and the constant channel speed with the above average speed, i.e., Eq.(35) and Eq.(36), to the solution found in [24]. 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 total solution time in the time-varying system.

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

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

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

Note that, the longer the time interval of the divisible job 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 will be 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 α's in the run time $t = 0$ to $t = 10$. The true job finish time occurs when the sum of these α's is equal to one by the normalization equation which is in this case between $t = 3.090$ and $t = 3.100$. Table 1 shows the results of the algorithm in section II.

Table 1.

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

Those 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.

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 2 shows the results of the exhaustive search.

Table 2.

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

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 with this constant processor speed and compare the result with the one from the solution of the non-time-varying system found in [24]. Table 3 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 4 shows the true results according to the solution in [24].

Table 3.

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 4.

T_f	α_1	α_2	α_3
2.049180	0.409836	0.327869	0.262295

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 3. If one chooses $T_f = 2.05$ in Table 3 as a solution of the algorithm, the accuracy is $(1 - \frac{2.05 - 2.049180}{2.049180}) \times 100\% \simeq 99.96\%$.

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 α's during the run time $t = 0$ to $t = 6$. The true job finish time occurs when the sum of α's 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 5 shows the results from the algorithm in section III and Table 6 shows the results from an exhaustive grid search in the solution space.

Table 5.

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 6.

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

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 is 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 [24]. Table 7 shows the results from the algorithm with constant channel speed and Table 8 shows the true results via [24].

Table 7.

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

Table 8.

T_f	α_1	α_2	α_3
2.049180	0.409836	0.327869	0.262295

All the true results (T_f, α_1, α_2 and α_3) lie between the two closest results from the algorithm in Table 7. The accuracy is the same as in the previous subsection and is approximately 99.96%.

C. Time-Varying Processor and Channel Speed

This subsection will briefly explain 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 9 shows the results from the algorithm in section IV and Table 10 shows the results from the exhaustive search.

Table 9.

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

Table 10.

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

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 11 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 12 shows the true results from [24].

Table 11.

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 12.

T_f	α_1	α_2	α_3
1.421053	0.473684	0.315789	0.210527

All the results (T_f, α_1, α_2 and α_3) lie between the two closest results from the algorithm in Table 11. If one chooses $T_f = 1.422$ in Table 11 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, Fig. 10 and Fig. 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 (α's) 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 Fig. 10 and Fig. 11, the legend is ordered in the order of the curves, i.e., the upper most 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 (α's) are not sensitive to the choice of queueing models, but they are sensitive with respect to the arrival rates (Fig. 10). 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 [illegible] ation 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 a 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.

Appendix A

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

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

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 $t_2 \le T_n < t_3$ and $x_f = 6$ since $t_6 \le T_f < t_7$. Then,

$$\begin{aligned}\int_{T_n}^{T_f} \frac{1}{w_n(t)}dt &= \frac{1}{w_n}\left[\sum_{k=0}^{x_n}\left(\sum_{j=0}^{k} r_j\right)^{-1}(T_f - T_n) + \sum_{k=x_n+1}^{x_f}\left(\sum_{j=0}^{k} r_j\right)^{-1}(T_f - t_k)\right] \\ &\quad - \frac{1}{w_n}\left[\sum_{k=0}^{x_n-1}\left(\sum_{j=0}^{k} r_j\right)^{-1}(T_f - T_n) + \sum_{k=x_n}^{x_f-1}\left(\sum_{j=0}^{k} r_j\right)^{-1}(T_f - t_{k+1})\right]\end{aligned}$$

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

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

$$\begin{aligned}
\frac{1}{w_n \sum_{j=0}^{x_f} r_j} &= \left.\frac{1}{w_n(t)}\right|_{t=T_f} = \frac{1}{w_n(T_f)} \\
\frac{1}{w_n \sum_{j=0}^{x_n} r_j} &= \left.\frac{1}{w_n(t)}\right|_{t=T_n} = \frac{1}{w_n(T_n)} \\
\frac{1}{w_n \sum_{j=0}^{k} r_j} &= \left.\frac{1}{w_n(t)}\right|_{t=t_k} = \frac{1}{w_n(t_k)} \\
\frac{1}{w_n \sum_{j=0}^{k-1} r_j} &= \left.\frac{1}{w_n(t)}\right|_{t=t_{k-1}} = \frac{1}{w_n(t_{k-1})}
\end{aligned}$$

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:

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

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

But,

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

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

Then,

$$\begin{aligned}
\int_{T_{n-1}}^{T_n}\frac{1}{Z(t)}dt &= \frac{1}{Z}\left[\sum_{k=0}^{x_{n-1}}\left(\sum_{j=0}^{k}s_j\right)^{-1}(T_n - T_{n-1}) + \sum_{k=x_{n-1}+1}^{x_n}\left(\sum_{j=0}^{k}s_j\right)^{-1}(T_n - t_k)\right] \\
&\quad - \frac{1}{Z}\left[\sum_{k=0}^{x_{n-1}-1}\left(\sum_{j=0}^{k}s_j\right)^{-1}(T_n - T_{n-1}) + \sum_{k=x_{n-1}}^{x_n-1}\left(\sum_{j=0}^{k}s_j\right)^{-1}(T_n - t_{k+1})\right] \\
&= \frac{T_n - T_{n-1}}{Z}\left[\sum_{k=0}^{x_{n-1}}\left(\sum_{j=0}^{k}s_j\right)^{-1} - \sum_{k=0}^{x_{n-1}-1}\left(\sum_{j=0}^{k}s_j\right)^{-1}\right] \\
&\quad + \frac{T_n}{Z}\left[\sum_{k=x_{n-1}+1}^{x_n}\left(\sum_{j=0}^{k}s_j\right)^{-1} - \sum_{k=x_{n-1}}^{x_n-1}\left(\sum_{j=0}^{k}s_j\right)^{-1}\right] \\
&\quad - \frac{1}{Z}\left[\sum_{k=x_{n-1}+1}^{x_n}\left(\sum_{j=0}^{k}s_j\right)^{-1}t_k - \sum_{k=x_{n-1}}^{x_n-1}\left(\sum_{j=0}^{k}s_j\right)^{-1}t_{k+1}\right] \\
&= \frac{T_n - T_{n-1}}{Z\sum_{j=0}^{x_{n-1}}s_j} + \frac{T_n}{Z}\left[\left(\sum_{j=0}^{x_n}s_j\right)^{-1} - \left(\sum_{j=0}^{x_{n-1}}s_j\right)^{-1}\right] - \frac{t_k}{Z}\sum_{k=x_{n-1}+1}^{x_n}\left[\left(\sum_{j=0}^{k}s_j\right)^{-1} - \left(\sum_{j=0}^{k-1}s_j\right)^{-1}\right] \\
&= \frac{T_n}{Z\sum_{j=0}^{x_n}s_j} - \frac{T_{n-1}}{Z\sum_{j=0}^{x_{n-1}}s_j} - \frac{t_k}{Z}\sum_{k=x_{n-1}+1}^{x_n}\left[\left(\sum_{j=0}^{k}s_j\right)^{-1} - \left(\sum_{j=0}^{k-1}s_j\right)^{-1}\right]
\end{aligned}$$

Note that $\left(Z\sum_{j=0}^{x_n}s_j\right)^{-1}$ 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,

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

Acknowledgement

The research in this paper was supported in part by the BMDO/IST under the U.S. Office of Naval Research under grant no. N00014-91-J4063.

REFERENCES

[1] S. H. Bokhari, "A network flow model for load balancing in circuit-switched multicomputers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 4, no. 6, pp. 649–657, June, 1993.

[2] V. M. Lo, S. Rajopadhye, S. Gupta, D. Keldsen, M. A. Mohamed, and J. Telle. "Mapping divide and conquer algorithms to parallel computers," In *Proceedings of the 1990 International Conference on Parallel Architectures*. 1990. pp. 128–135.

[3] K. Ramamrithamm, J. A. Stankovic, and P.-F. Shiah, "Efficient scheduling algorithms for real-time multiprocessor systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 1, no. 2, pp. 184–194, April, 1990.

[4] Y.-C. Chang and K. G. Shin, "Optimal load sharing in distributed real-time systems," *Journal of Parallel and Distributed Computing*, vol. 19, no. 1, pp. 38–50, September, 1993.

[5] K. G. Shin and M.-S. Chen, "On the number of acceptable task assignments in distributed computing systems," *IEEE Transactions on Computers*, vol. 39, no. 1, pp. 99–110, January, 1990.

[6] D.-T. Peng and K. G. Shin, "A new performance measure for scheduling independent real-time tasks," *Journal of Parallel and Distributed Computing*, vol. 19, no. 1, pp. 11–26, September, 1993.

[7] C.-H. Lee, D. Lee, and M. Kim, "Optimal task assignment in linear array networks," *IEEE Transactions on Computers*. vol. 41, no. 7, pp. 877–880, July, 1992.

[8] G. C. Sih and E. A. Lee, "Declustering: A new multiprocessor scheduling technique," *IEEE Transactions on Parallel and Distributed Systems*, vol. 4, no. 6, pp. 625–637, June, 1993.

[9] K. K. Goswami, M. Devarakonda, and R. K. Iyer, "Prediction-based dynamic load-sharing heuristics," *IEEE Transactions on Parallel and Distributed Systems*, vol. 4, no. 6, pp. 638–648, June, 1993.

[10] J. Xu and K. Hwang, "Heuristic methods for dynamic load balancing in a message-passing multicomputer," *Journal of Parallel and Distributed Computing*, vol. 18, no. 1, pp. 1–13, May, 1993.

[11] X. Qian and Q. Yang, "An analytical model for load balancing on symmetric multiprocessor systems," *Journal of Parallel and Distributed Computing*, vol. 20, pp. 198–211, 1994.

[12] I. Ahmad, A. Ghafoor, and G. C. Fox, "Hierarchical scheduling of dynamic parallel computations on hypercube multicomputers," *Journal of Parallel and Distributed Computing*, vol. 20, pp. 317–329, 1994.

[13] G. Huang and W. Ongsakul, "An efficient load-balancing processor scheduling algorithm for parallelization of gauss-seidel type algorithms," *Journal of Parallel and Distributed Computing*, vol. 22, pp. 350–358, 1994.

[14] J. Blazewicz, M. Drabowski, and J. Weglarz, "Scheduling multiprocessor tasks to minimize schedule length," *IEEE Transactions on Computers*, vol. C-35, pp. 389–398, May, 1986.

[15] J. Du and J. Y.-T. Leung, "Complexity of scheduling parallel task systems," *SIAM Journal on Discrete Mathematics*. vol. 2, pp. 473–487, November, 1989.

[16] W. Zhao, K. Ramamritham, and J. A. Stankovic, "Preemptive scheduling under time and resource constraints," *IEEE Transactions on Computers*, vol. C-36, pp. 949–960, August, 1987.

[17] Y. C. Cheng and T. G. Robertazzi, "Distributed computation with communication delays," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 24, no. 6, pp. 700–712, November, 1988.

[18] Y. C. Cheng and T. G. Robertazzi, "Distributed computation for a tree network with communication delays," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 26, no. 3, pp. 511–516, May, 1990.

[19] S. Bataineh and T. G. Robertazzi, "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. March, 1991 pp. 709–714.

[20] S. Bataineh and T. G. Robertazzi, "Bus oriented load sharing for a network of sensor driven processors," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 21, no. 5, pp. 1202–1205, September, 1991.

[21] S. Bataineh and T. G. Robertazzi, "Ultimate performance limits for networks of load sharing processors," In *Proceedings of the 1992 Conference on Information Science and Systems*, Princeton University, Princeton, NJ. March, 1992, pp. 794–7[illegible]

[22] S. Bataineh, T. Hsiung, and T. G. Robertazzi, "Closed form solutions for bus and tree networks of processors load sharing a divisible job," *IEEE Transaction on Computers*, vol. 43, no. 10, pp. 1184–1196, October, 1994.

[23] T. G. Robertazzi. "Processor equivalence for a linear daisy chain of load sharing processors," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 29. no. 4. pp. 1216–1221, October. 1993.

[24] J. Sohn and T. G. Robertazzi, "Optimal divisible job load sharing for bus networks," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 32, no. 1,, January, 1996.

[25] J. Sohn and T. G. Robertazzi, "Optimal load sharing for a divisible job on bus network," In *Proceedings of the 1993 Conference on Information Science and Systems*, The Johns Hopkins University, Baltimore, MD. March, 1993.

[26] J. Sohn and T. G. Robertazzi, "A multi-job load sharing strategy for divisible job on bus networks," *SUNY at Stony Brook College of Engineering and Applied Science Technical Report*, no. 697, August, 1994.

[27] H. J. Kim, G. I. Jee, and J. G. Lee, "Optimal load distribution for tree network processors," submitted for publication.

[28] D. Ghose and V. Mani, "Distributed computation in a linear network: Closed-form solutions and computational techniques," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 30, no. 2, pp. 471–483, April, 1994.

[29] D. Ghose and V. Mani, "Distributed computation with communication delays: Asymptotic performance analysis," *Journal of Parallel and Distributed Computing*, November, 1994.

[30] V. Bharadwaj, D. Ghose, and V. Mani, "Optimal sequencing and arrangement in distributed single-level tree networks with communication delays," *IEEE Transactions on Parallel and Distributed Systems*, vol. 5, no. 9, pp. 968–976, September, 1994.

[31] V. Bharadwaj, D. Ghose, and V. Mani, "Installment techniques in tree networks," accepted by the IEEE Transactions on Aerospace and Electronic System.

[32] V. Bharadwaj, D. Ghose, and V. Mani, "An efficient load distribution strategy for a distributed linear network of processors with communication delays," accepted by the Computer and Mathematics with Applications.

Figure Captions

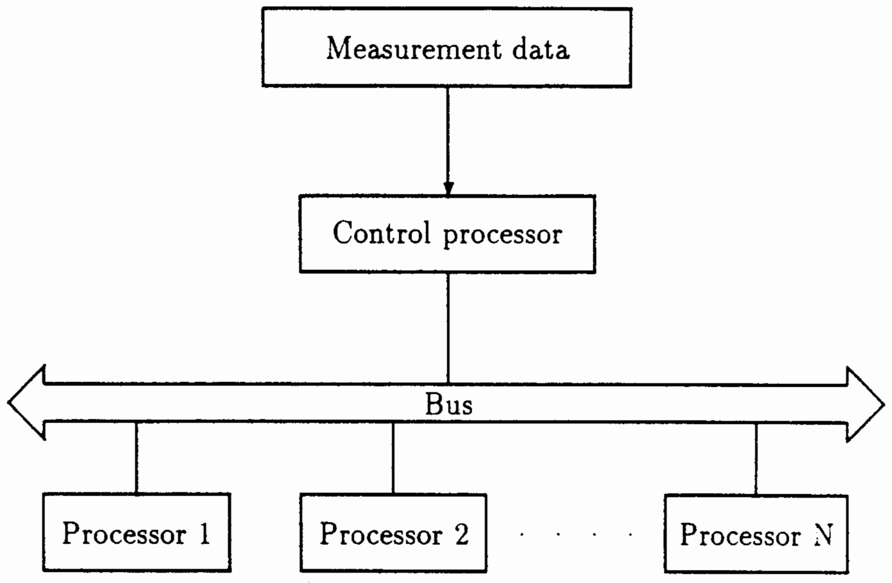


Fig. 1. **Bus network with** load originationa at a control processor.

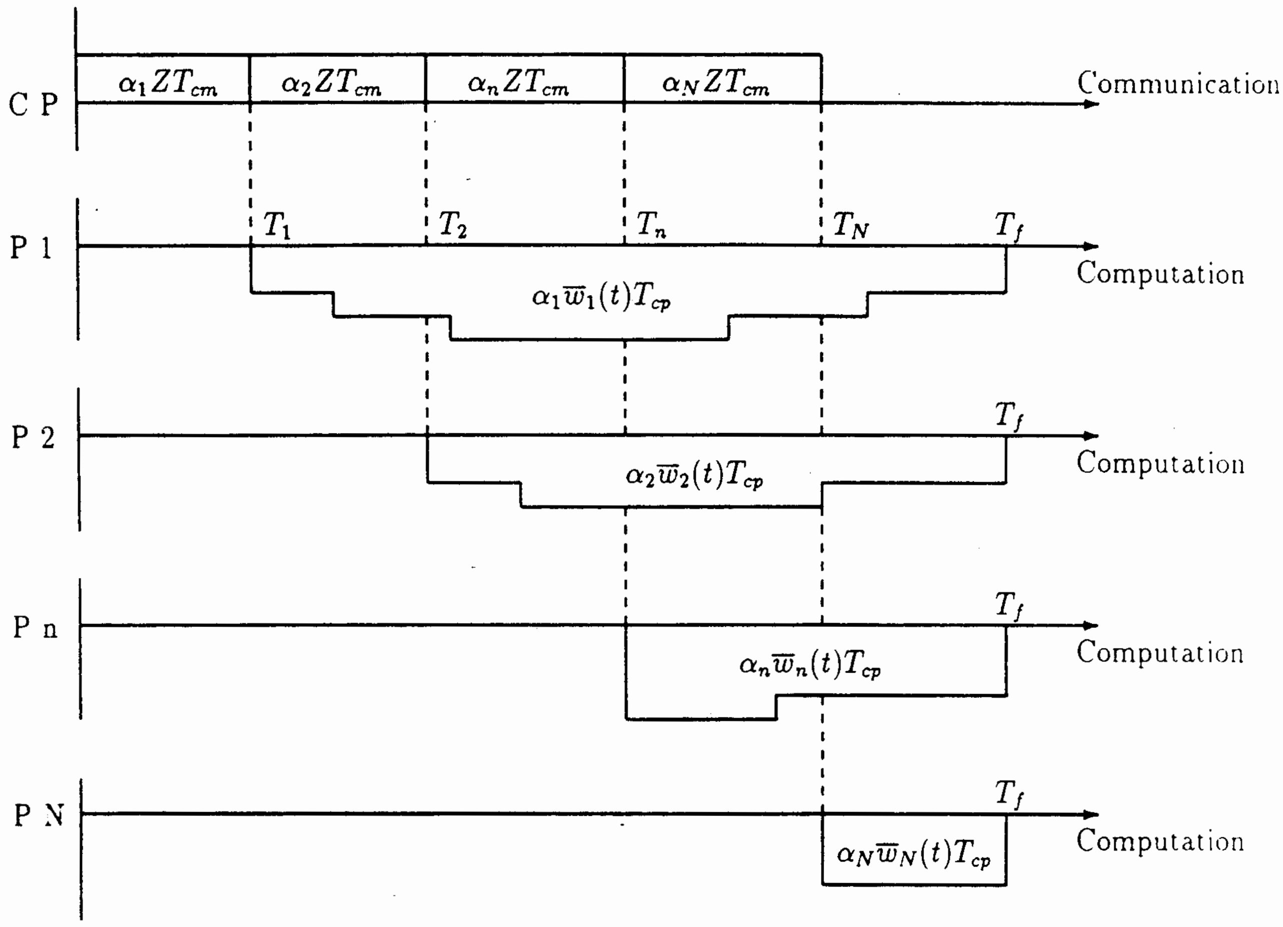


Fig. 2. Timing diagram for the bus network with time-varying processor speed.

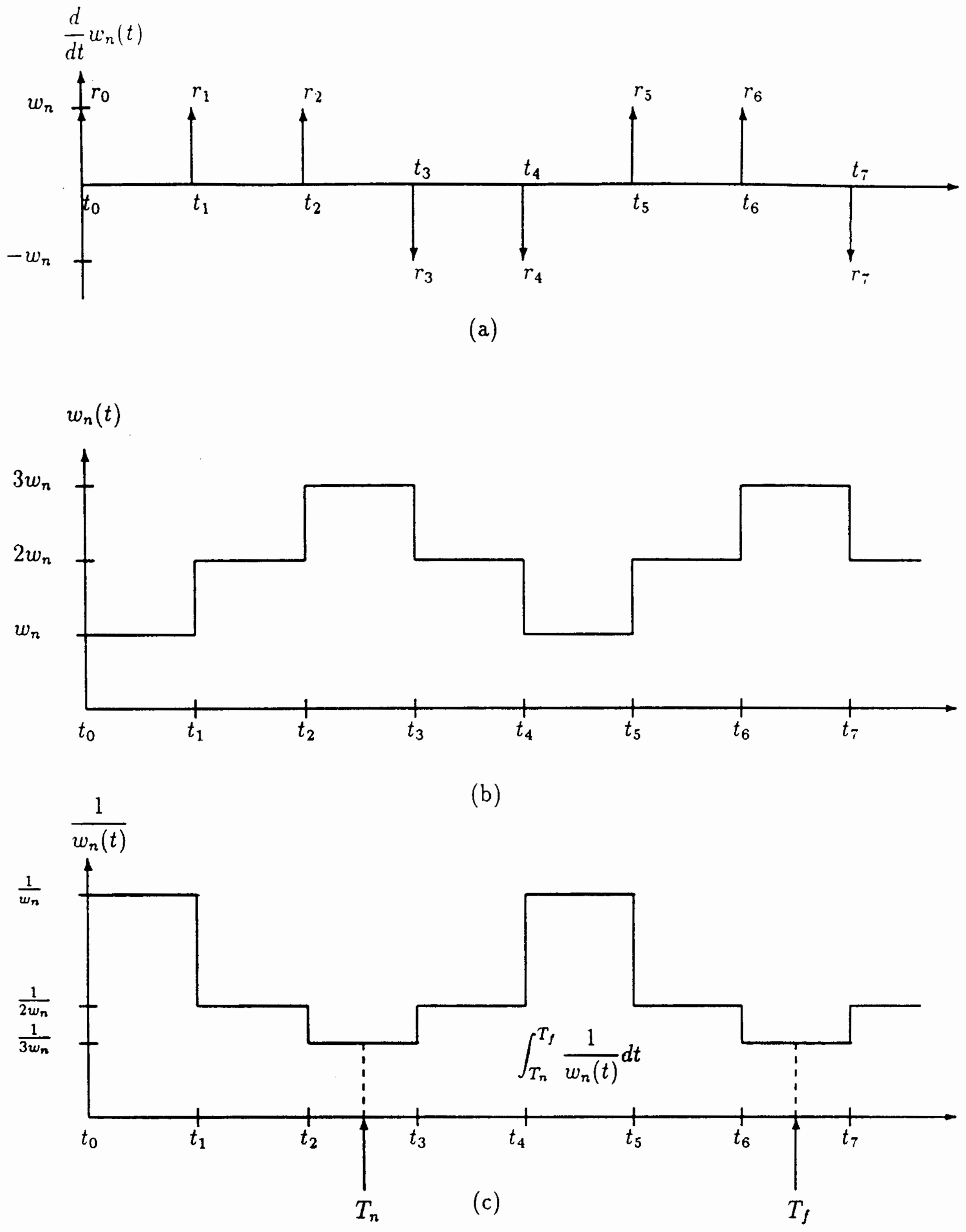


Fig. 3. (a) The derivative of the timing process which is inversely proportional to the computing speed (b) The timing process which is inversely proportional to the computing speed (c) The timing process which is proportional to the computing speed.

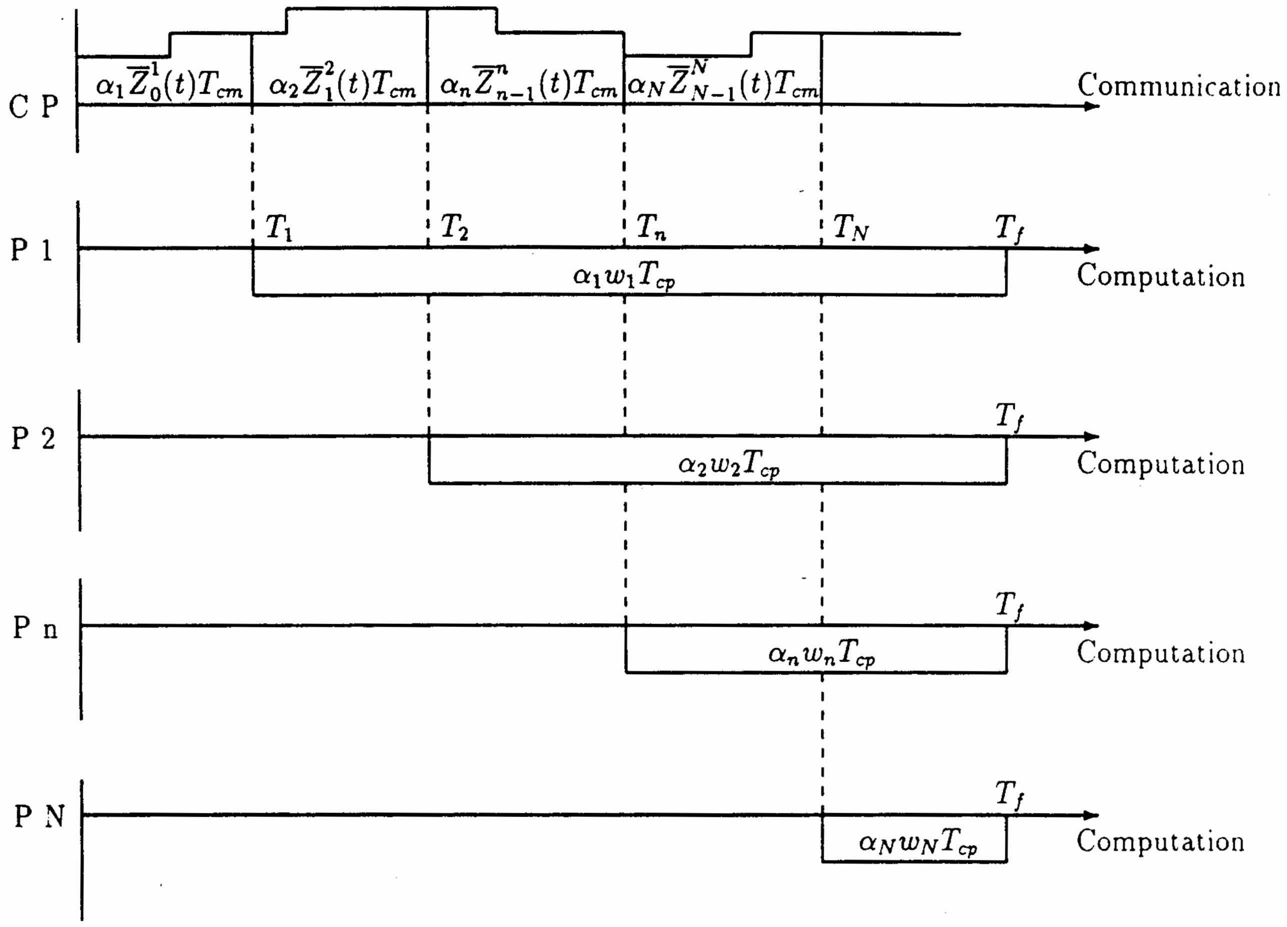
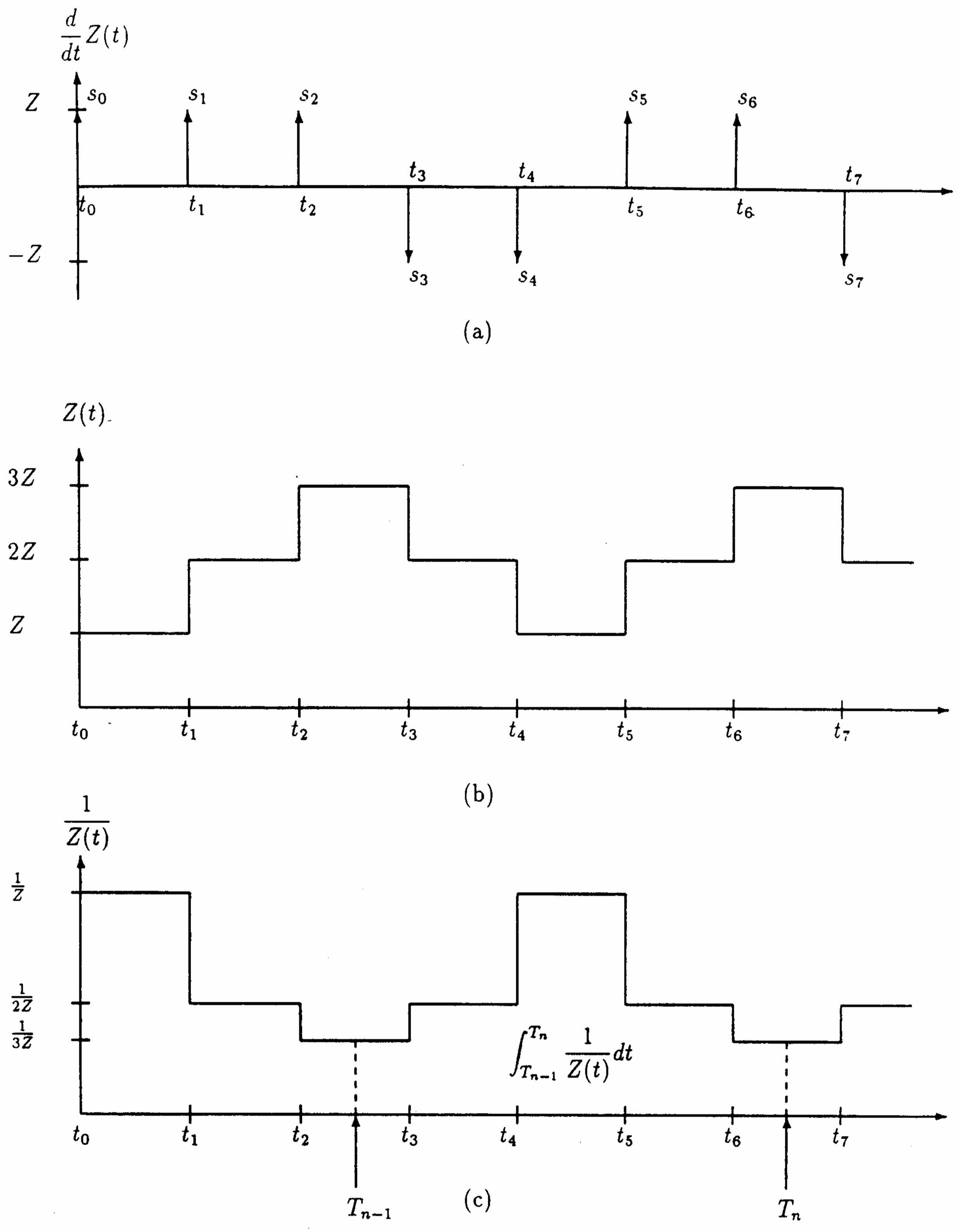


Fig. 4. Timing diagram for the bus network with time-varying channel speed.



$\frac{d}{dt}Z(t)$
Z
$-Z$
s_0 s_1 s_2 s_3 s_4 s_5 s_6 s_7
t_0 t_1 t_2 t_3 t_4 t_5 t_6 t_7
(a)
$Z(t)$
$3Z$
$2Z$
Z
t_0 t_1 t_2 t_3 t_4 t_5 t_6 t_7
(b)
$\frac{1}{Z(t)}$
$\frac{1}{Z}$
$\frac{1}{2Z}$
$\frac{1}{3Z}$
$\int_{T_{n-1}}^{T_n} \frac{1}{Z(t)} dt$
t_0 t_1 t_2 t_3 t_4 t_5 t_6 t_7
T_{n-1}
T_n
(c)

Fig. 5. (a) The derivative of the timing process which is inversely proportional to the channel speed (b) The timing process which is inversely proportional to the channel speed (c) The timing process which is proportional to the channel speed.

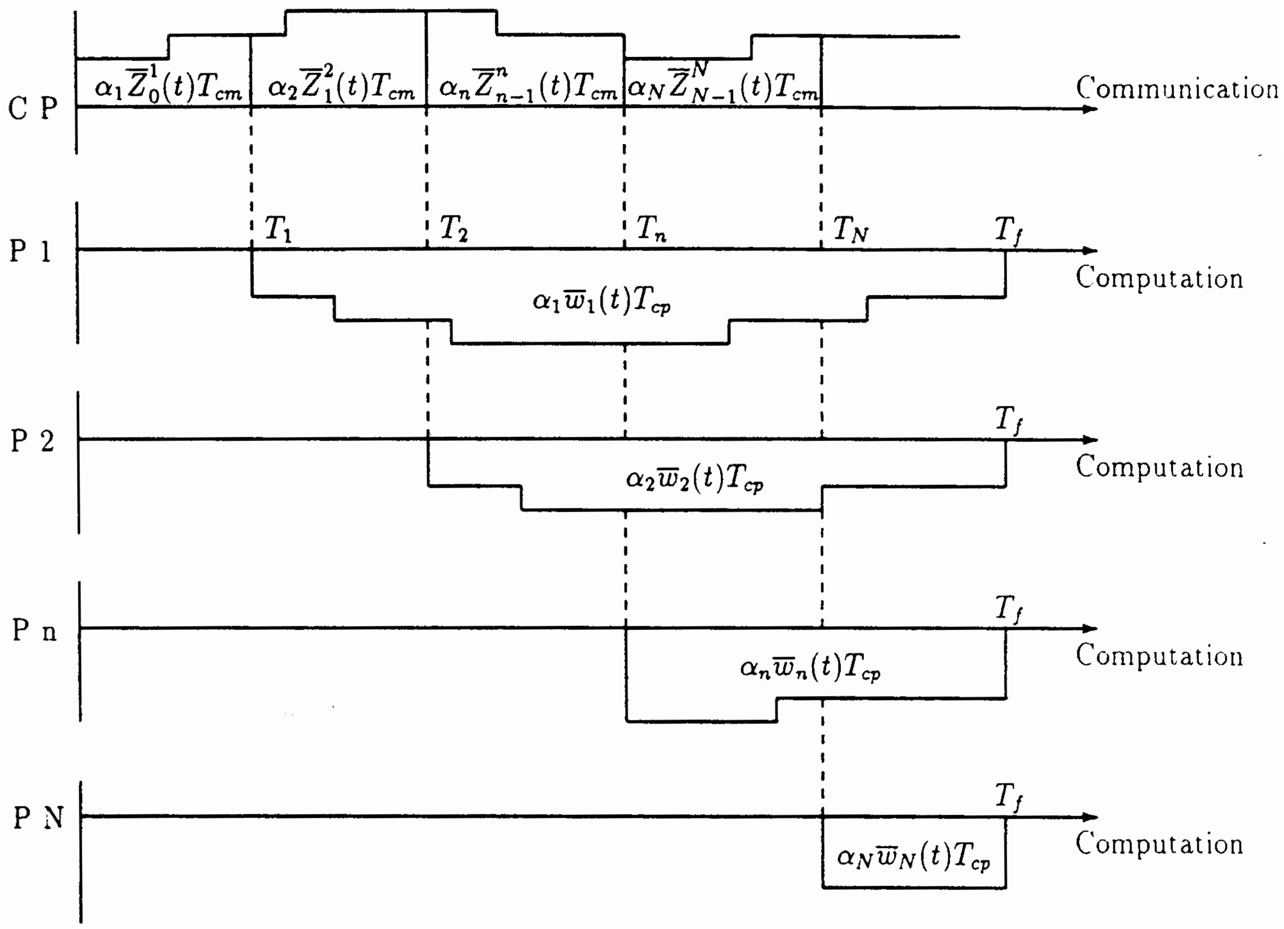

C P
$\alpha_1\overline{Z}_0^1(t)T_{cm}$
$\alpha_2\overline{Z}_1^2(t)T_{cm}$
$\alpha_n\overline{Z}_{n-1}^n(t)T_{cm}$
$\alpha_N\overline{Z}_{N-1}^N(t)T_{cm}$
Communication
P 1
T_1
T_2
T_n
T_N
T_f
$\alpha_1\overline{w}_1(t)T_{cp}$
Computation
P 2
$\alpha_2\overline{w}_2(t)T_{cp}$
Computation
P n
$\alpha_n\overline{w}_n(t)T_{cp}$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

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

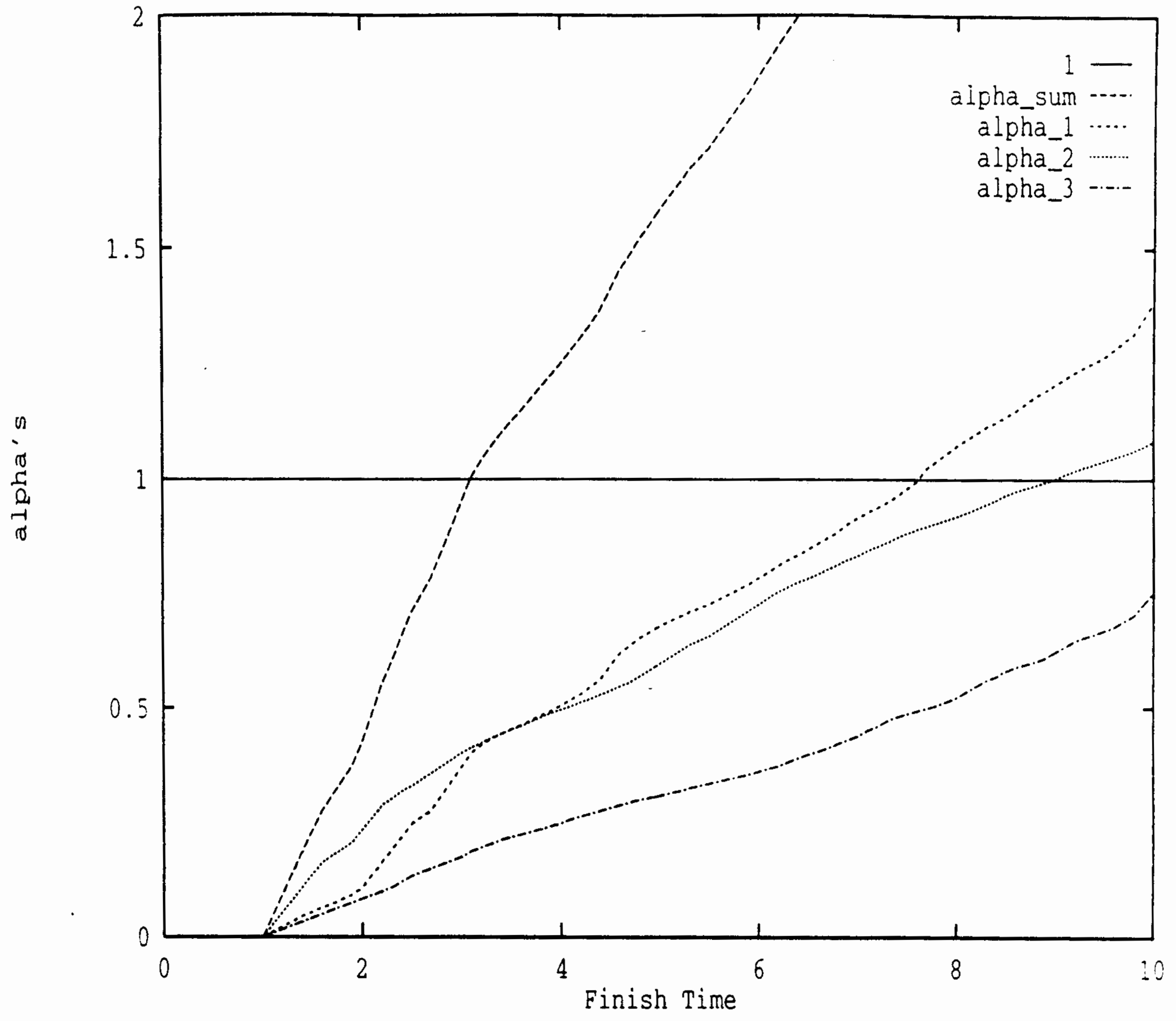


Fig. 7. Job computing finish time for time-varying processor speed.

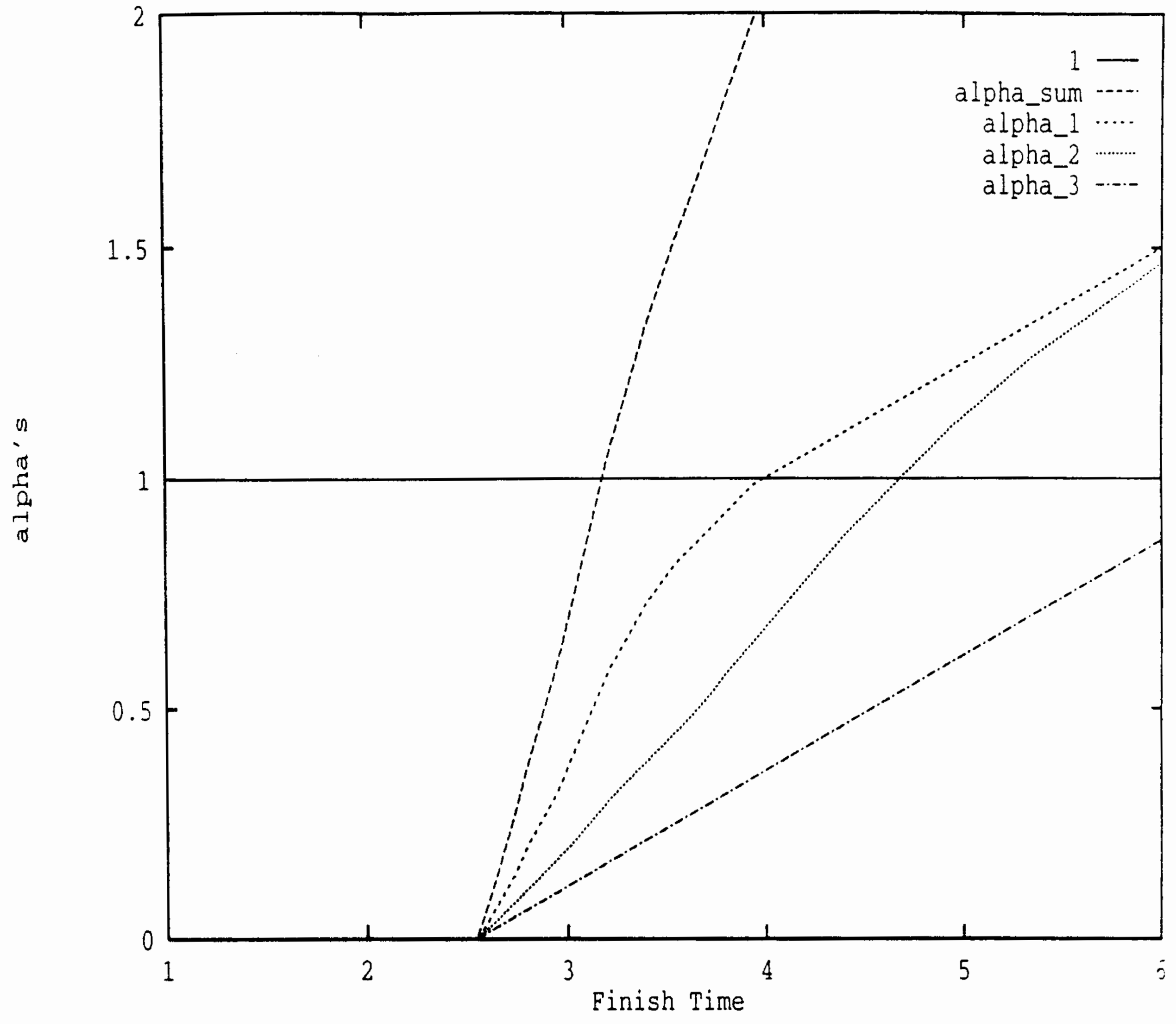


Fig. 8. Job computing finish time for time-varying channel speed.

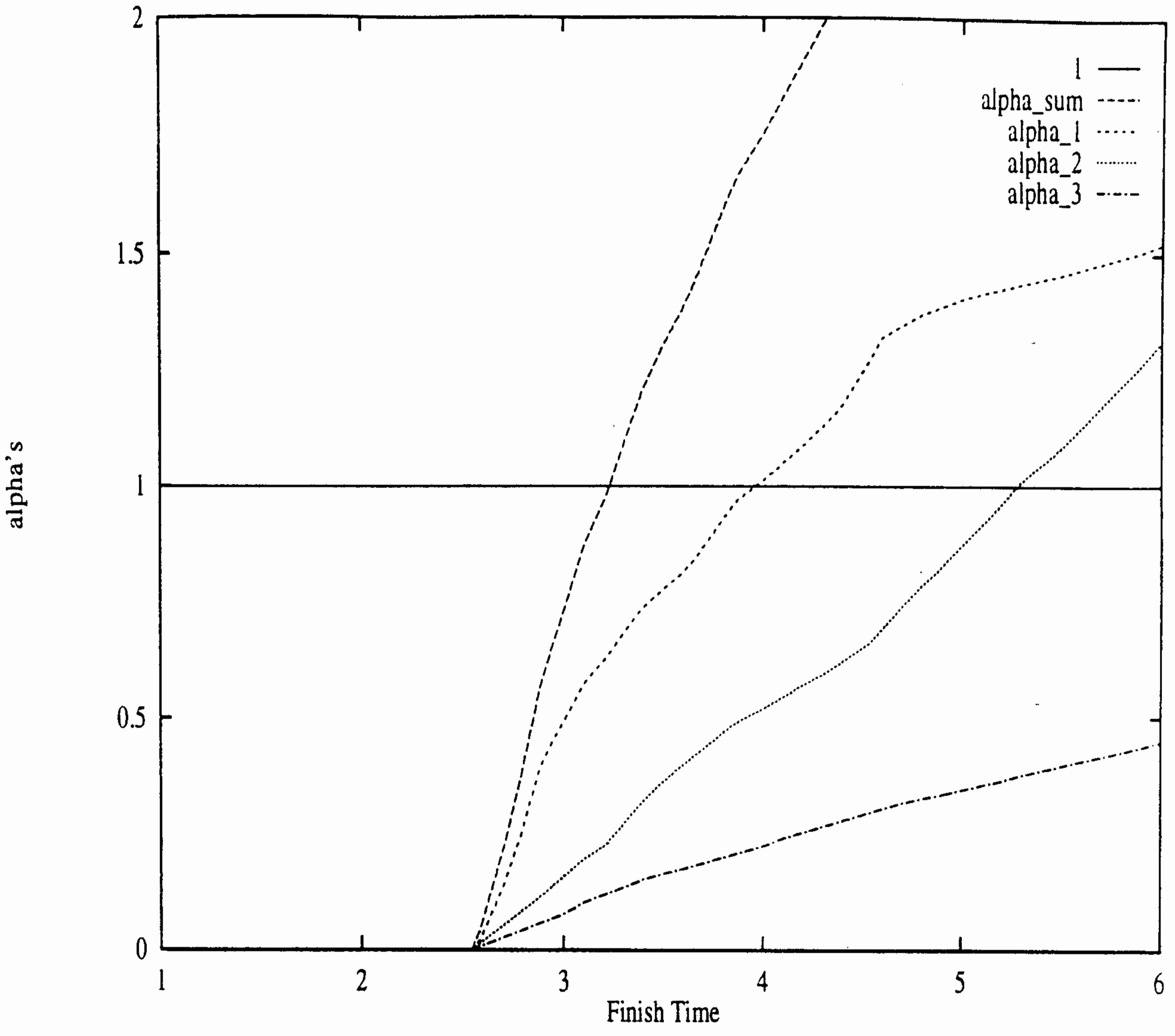


Fig. 9. Job computing finish time for time-varying processor speed and channel speed.

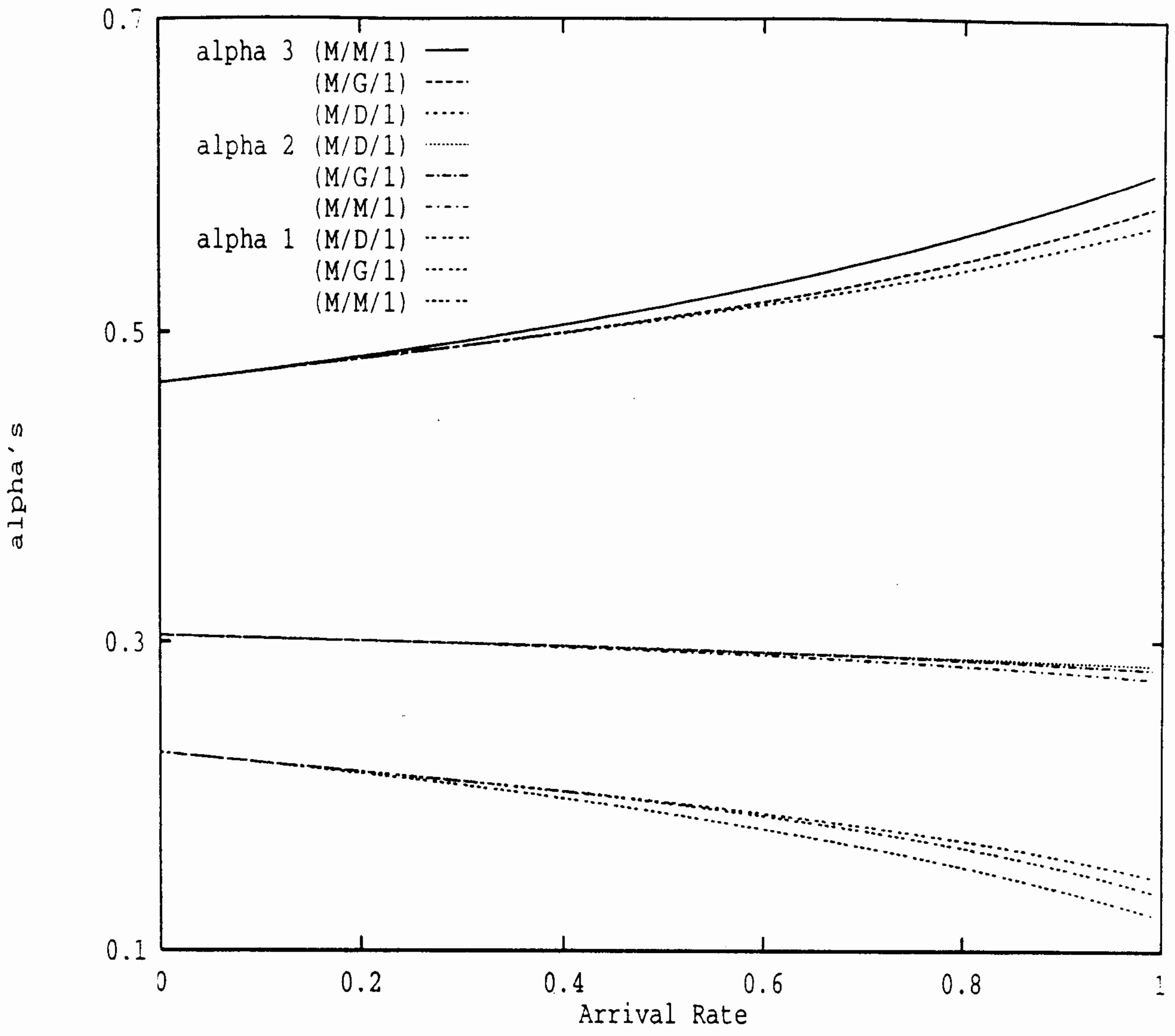


Fig. 10. The optimal fraction of the workload (α's) vs. arrival rate in the stochastic analysis.

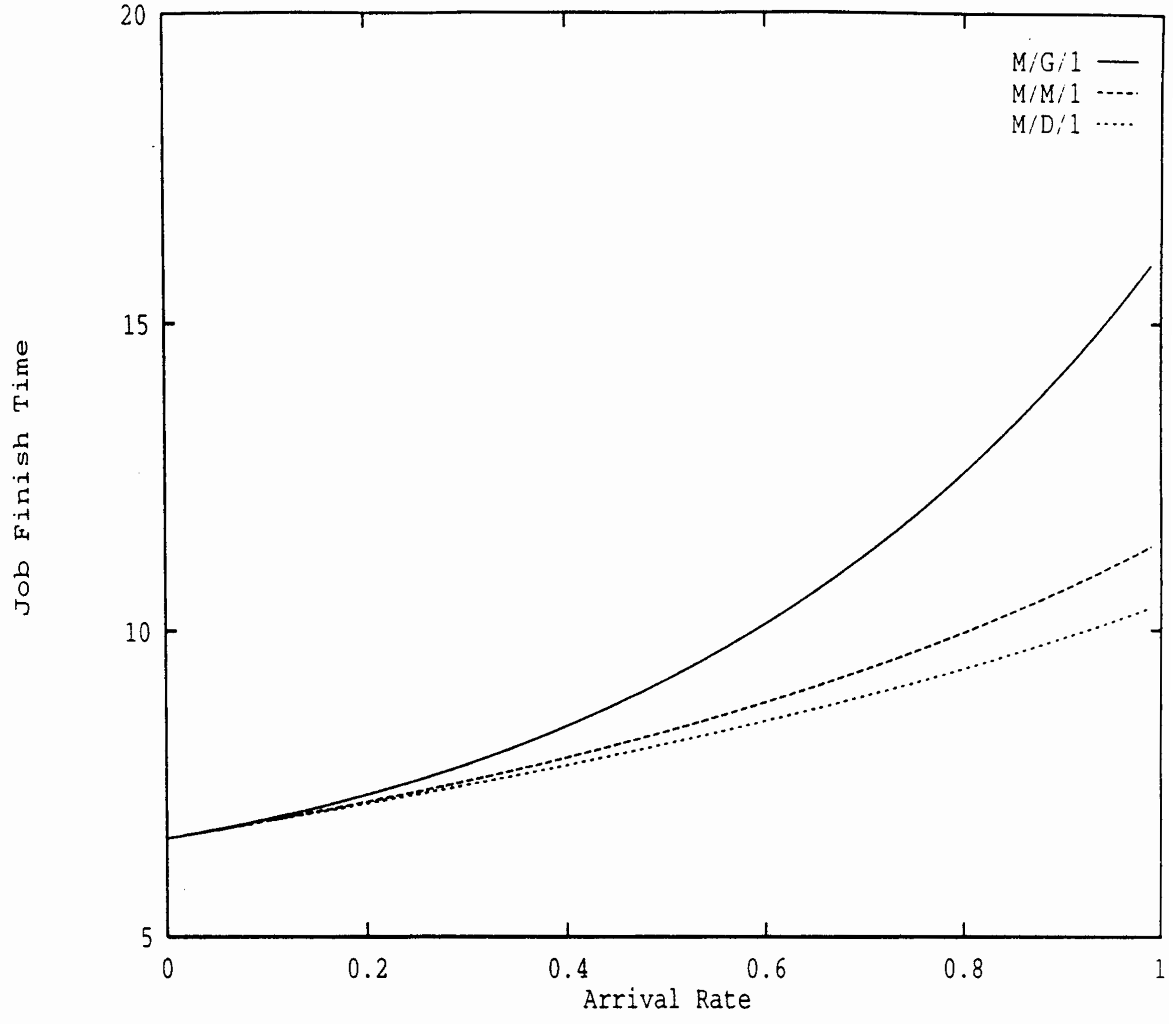


Fig. 11. Job computing finish time in the stochastic analysis.

Affiliation of Authors:

Jeeho Sohn and Prof. Thomas G. Robertazzi

Dept. of Electrical Engineering,

SUNY at Stony Brook,

Stony Brook, NY 11794

(516) 632-8412/8400

tom@sbee.sunysb.edu

Address all correspondence to T. Robertazzi.

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 at Boulder in 1989. He is presently a Ph.D candidate in the electrical engineering department at the University at Stony Brook. 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 in 1981 and the B.E.E. from the Cooper Union in 1977. He is presently an associate professor of electrical engineering at the University at Stony Brook. During 1982-83 he was an assistant professor in the electrical engineering department of Manhattan College, Riverdale N.Y. Prof. Robertazzi is currently editor for books for the IEEE Communications Society and an associate editor of the journal Wireless Networks. His research interests involve the performance evaluation of computer and communication systems.