JOURNAL OF PARALLEL AND DISTRIBUTED COMPUTING 23, 293–305 (1994)

Distributed Computation with Communication Delays: Asymptotic Performance Analysis

D. GHOSE AND V. MANI

Department of Aerospace Engineering, Indian Institute of Science, Bangalore 560 012, India

This paper analyzes the effect of communication delay on the optimal distribution of processing loads in distributed computing networks. The processing load is assumed to satisfy the property of arbitrary divisibility. The objective is to divide and distribute this processing load among various processors in the network in order to minimize the processing time. An asymptotic analysis of the performance of such networks is carried out to obtain a limit on the performance enhancement obtained by using additional processors. The architectures considered are linear and single-level tree configurations. The cases when the processors are equipped with and without front-ends are considered. © 1994 Academic Press, Inc.

1. INTRODUCTION

One of the objectives in a distributed computing network is to obtain an optimal distribution of processing load among processors in order to achieve minimum processing time. The processing load may be of two types, namely, indivisible and arbitrarily divisible. A load is said to be indivisible if it cannot be further divided and has to be processed in its entirety. It is arbitrarily divisible if, for all practical purposes, it can be divided into infinitesimally small fractions and then distributed among the processors. The optimal distribution of an indivisible load has received much attention in the literature [5, 10, 11], whereas studies on the distribution of an arbitrarily divisible load is recent [6, 7]. Divisible loads are characterized by their large volume, in which each data element requires exactly the same type of processing. This kind of processing load occurs typically in a radar tracking or an image processing system, or in the processing of massive experimental data. One of the major problems which plagues such systems is that of communication delay between processors, which effectively limits the speed-up that can be achieved by using a large number of processors [3]. Problems of this nature were considered by Cheng and Robertazzi [6, 7] and Bataineh and Robertazzi [1] for linear, tree, and bus-oriented networks of communicating processors. In these papers, a basis for an optimal strategy for load distribution was also presented. The computational results in these papers showed that increasing the number of processors beyond a certain value did not improve the performance of the network significantly. This was due to the deleterious effect of communication delay. However, although the above results were intuitively convincing, a considerable amount of computation was required to observe this phenomena. Bataineh and Robertazzi [2] also obtained the ultimate performance limits for linear and multilevel tree networks using the concept of processor equivalence. Recently, the authors obtained closed-form solutions to the optimal load distribution problem in such distributed networks [4, 9] and analytically proved a number of results which were demonstrated only through computational means in earlier studies.

In this paper, we present an asymptotic analysis of the effect of communication delay on linear and tree distributed computing networks. We show that as in the case of Amdahl's law the presence of the sequential portion of the algorithm limits the utility of parallelism [12], in these cases the presence of communication delay also has a similar effect on load distribution.

2. BASIC CONCEPTS

The network consists of many processors connected to each other through communication channels. Each processor, upon receiving a computational load, distributes it among its neighbors. A processor starts computing only after it receives the whole of the load assigned to it. The processors may or may not be equipped with front-ends. If a processor is equipped with a front-end, it can transmit computational load to its neighbors using the front-end while it simultaneously computes the load it has kept for itself. However, a processor without a front-end cannot perform these two tasks simultaneously. We make an important assumption here that a processor can receive computational load from only one preceding processor, while it can distribute load to many succeeding processors; i.e., each processor has at most one incoming communication channel, while it may have many outgoing communication channels. Another important assumption we make here is that the load can be distributed in any desired fraction of the total load assigned to a distributed computing network. Thus, we define $\alpha \in [0, 1]$ as the fraction of the total load assigned to a processor.

The time taken by a processor to compute a fraction α of the total computing load is given by $\alpha w T_{cp}$, where w is

0743-7315/94 $6.00

inversely proportional to the computing speed of the processor and T_{cp} is the time taken by a standard processor to process the total computational load. Similarly, the time delay in transmitting a fraction a of the computational load from one processor to another is $\alpha z T_{cm}$, where z is inversely proportional to the speed of the channel and T_{cm} is the time delay to transmit the total computational load through a standard communication channel. Obviously, for a standard processor and a standard channel, $w = 1$ and $z = 1$, respectively. When $w = a$ (or $z = a$) the processor (or the channel) is $1/a$ times as fast as the standard processor (or standard channel). In general, the communication delay in a channel is modeled as the sum of two parts, one of which is a constant and is independent of the amount of load carried by the channel, the other of which is linearly proportional to the amount of load [3]. In many networks the first part is negligibly small and therefore in this paper we also choose to neglect it.

The time T_N taken by a distributed computing network having $N + 1$ processors, indexed by $i = 0, 1, \ldots, N$, to completely process a given computational load is given by

$$T_N = \max (t_0, t_1, \ldots, t_N), \tag{1}$$

where t_i is the time that elapses between the beginning of the process at $t = 0$ (when one of the processors in the network, called the root processor and indexed by $i = 0$, receives the total computational load and initiates the process) and the time when the ith processor ceases operation. The problem faced by the designer is one of load allocation to individual processors so that this time T_N is minimized. Cheng and Robertazzi [6] have proved that, to minimize T_N, all the processors must stop computing at the same instant in time.

This value of T_N can be further reduced if we add more processors to the distributed network, but it is seen from the computational results in [1, 6] that the value of T_N levels off after only a few processors. However, analytical proof of these results remain elusive. Our objective in this paper is to obtain these analytical results for essentially two important classes of networks with linear and tree architectural configurations. In the linear network, we consider the case in which the load originates at the boundary of the network. In the tree network, we consider only the single-level tree. In both kinds of networks, we consider processors equipped with and without front-ends.

3. LINEAR NETWORK WITH FRONT-END

In this section, we shall consider linear networks in which all the processors are equipped with front-ends. We shall assume that all communication channels have the same speed (i.e., equal z) and all processors have the same speed (i.e., equal w).

3.1. Basic Model and Closed-Form Expressions

Let the linear network be as shown in Fig. 1, which also shows the timing diagram of the load transfer and computational process. The load-sharing equations are as follows:

$$\alpha_i w T_{cp} = (1 - \alpha_0 - \alpha_1 - \cdots - \alpha_i) z T_{cm} + \alpha_{i+1} w T_{cp}, \quad i = 0, 1, \ldots, N - 1. \tag{2}$$

The normalizing equation is

$$\alpha_0 + \alpha_1 + \cdots + \alpha_N = 1. \tag{3}$$

Equation (2) can be rewritten as

$$\alpha_i = (1 - \alpha_0 - \alpha_1 - \cdots - \alpha_i)\tau + \alpha_{i+1}, \quad i = 0, 1, \ldots, N - 1, \tag{4}$$

where

$$\tau = (z/w)(T_{cm}/T_{cp}). \tag{5}$$

Equations (3) and (4) can be solved by expressing all α_i, $i = 0, 1, \ldots, N - 1$, in terms of α_N and substituting in (3). This operation yields the following closed-form solution derived by Mani and Ghose [9]:

$$\alpha_j = f_j(N, \tau) / \sum_{j=0}^{N} f_j(N, \tau). \tag{6}$$

Here

$$f_j(N, \tau) = c(j, 0)\tau^0 + c(j, 1)\tau^1 + \cdots + c(j, 1)\tau^i + \cdots + c(j, N - j)\tau^{N-j}. \tag{7}$$

The function $f_j(N, \tau)$ is a polynomial of $(N - j)$ degree and the coefficient of τ^i in the function $f_j(N, \tau)$ is given by

$$c(j, i) = \{(N - j + i)!\}/\{(2i)!\,(N - j - i)\}, \quad j = 0, 1, \ldots, N, \; i \le (N - j). \tag{8}$$

The difference between $f_j(N, \tau)$ defined here and $f_j(\tau)$ defined in our earlier study [9] is that in our earlier study the processors were numbered from the last processor to the starting/root processor. In this study, the processors are numbered from the root processor to the last processor. Also, in the earlier study, the total number of processors in the network was N, whereas in this study it is $(N + 1)$. Equations (6)–(8) are sufficient to obtain the optimal load distribution among $(N + 1)$ processors in a given linear network of distributed processors with front-ends. However, we shall not make explicit use of the closed-form expression in the ensuing analysis. The time taken to complete a computational task assigned to a network

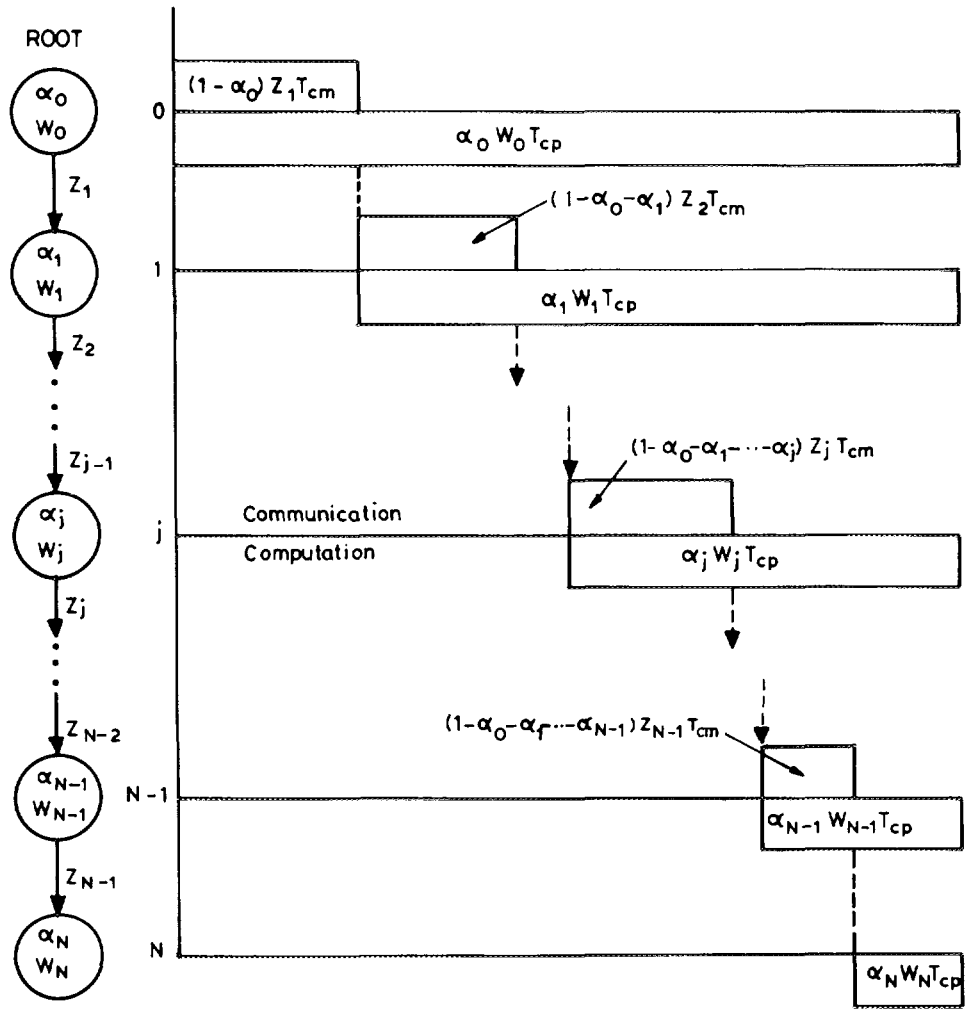

FIG. 1. Timing diagram for a linear network with front-end.

with N processors connected to the root processor in a linear array is given by

$$T_N = wT_{cp} f_0(N, \tau) / \sum_{j=0}^{N} f_j(N, \tau). \tag{9}$$

Note that $T_0 = wT_{cp}$ is the time taken by the root processor alone to process the entire load.

Consider such a network with N processors attached to the root. Let the parameters z and w be the same for all channels and processors. In order to improve the performance of the linear network (i.e., minimize T_N), we can connect additional processors in a linear array. Let $S(N, N + n)$ denote the speed-up achieved by adding n processors in a linear fashion to the existing N processors. Then

$$S(N, N + n) = T_N / T_{N+n}, \tag{10}$$

which can be written as

$$S(N, N + n) = (T_0 / T_{N+n}) / (T_0 / T_N). \tag{11}$$

Hence

$$S(N, N + n) = S(0, N + n) / S(0, N), \tag{12}$$

with $S(0, 0) = 1$. As expected, (Eq. 12) displays the semigroup property. Also,

$$S(0, N) = \sum_{j=0}^{N} f_j(N, \tau) / f_0(N, \tau), \tag{13}$$

which is the speed-up achieved by using N extra processors (having the same processing speed), in a linear array, over the root processor.

Another useful performance measure is the fractional savings in computation time when n additional processors are added to the existing N processors. The percentage savings in computation time can be obtained by multiplying the fractional savings in computation time by 100. This fractional savings in computation time, denoted $P(N, N + n)$, is given by

$$P(N, N + n) = (T_N - T_{N+n}) / T_N. \tag{14}$$

Using (10) and (12), the above expression reduces to

$$P(N, N + n) = 1 - S(0, N)/S(0, N + n). \quad (15)$$

The performance measure $S(N, N + n)$ is important in the case of those networks which process large quantities of load arriving at frequent intervals. Then $S(N, N + n)$ denotes the factor of increase in the amount of load which can be processed by the network in a given interval of time. The performance measure $P(N, N + n)$ is important in those networks whose computational output is required elsewhere for further processing. Then $P(N, N + n)$ denotes the savings in time in providing the required output.

3.2. Asymptotic Analysis

Here we examine the behavior of $S(N, N + n)$ and $P(N, N + n)$ as $n \to \infty$. Consider a linear network with m processors connected to the root processor. Substituting $i = 0$ in (4), we obtain

$$\alpha_0 = (1 - \alpha_0)\tau + \alpha_1. \quad (16)$$

Substituting (6), the above expression can be rewritten as

$$\sum_{j=0}^{m} f_j(m, \tau) = f_0(m, \tau) + (1/\tau)\{f_0(m, \tau) - f_1(m, \tau)\} \quad (17)$$

Using the above, $S(0, m)$ from Eq. (13) can be rewritten for $m > 1$ as

$$S(0, m) = 1 + (1/\tau)\{1 - f_1(m, \tau)/f_0(m, \tau)\}. \quad (18)$$

Thus

$$\lim_{m\to\infty} S(0, m) = 1 + (1/\tau)\{1 - \lim_{m\to\infty} f_1(m, \tau)/f_0(m, \tau)\} \quad (19)$$

and is denoted $S(0, \infty)$. Now, substituting $i = k$ and $i = k + 1$ in (4), using (6), and performing some simple algebraic manipulations, we obtain the following recursive relationship:

$$f_{k+1}(m, \tau)/f_k(m, \tau) = 1/\{(2 + \tau) - (f_{k+2}(m, \tau)/f_{k+1}(m, \tau)\}. \quad (20)$$

Thus

$$\lim_{m\to\infty} f_1(m, \tau)/f_0(m, \tau) = 1/\{(2 + \tau) - 1/\{(2 + \tau) - 1/\{(2 + \tau) - \ldots, \quad (21)$$

which is in the form of a continued fraction [8] and can be easily solved to yield

$$\lim_{m\to\infty} f_1(m, \tau)/f_0(m, \tau) = 1 + (1/2)\{\tau - \sqrt{\tau(\tau + 4)}\}, \quad (22)$$

from which

$$S(0, \infty) = (1/2)\{\sqrt{1 + (4/\tau)}\}. \quad (23)$$

The variation in $S(0, m)$ with m can be obtained from the following recursive relationship which can be obtained by substituting (20) into (18):

$$S(0, m) = \{1 + (\tau + 1)S(0, m - 1)\}/\{1 + \tau S(0, m - 1)\}. \quad (24)$$

Here $S(0, 0) = 1$. From (12),

$$\lim_{n\to\infty} S(N, N + n) = \{1/S(0, N)\} \lim_{n\to\infty} S(0, N + n). \quad (25)$$

We define

$$\lim_{n\to\infty} S(N, N + n) = S(N, \infty), \quad (26)$$

and rewrite (25) as

$$S(N, \infty) = S(0, \infty)/S(0, N), \quad (27)$$

where $S(0, N)$ can be calculated using (24). Similarly, denoting

$$\lim_{n\to\infty} P(N, N + n) = P(N, \infty), \quad (28)$$

we get

$$P(N, \infty) = \{1 - S(0, N)/S(0, \infty)\}, \quad (29)$$

from which we obtain

$$P(0, \infty) = \{1 + (\tau/2)(1 - \sqrt{1 + 4/\tau})\}. \quad (30)$$

Using (24), we can also derive the recursive relation

$$P(0, m) = 1/\{(2 + \tau) - P(0, m - 1)\}. \quad (31)$$

Let α_j^N denote the computational load assigned to the jth processor when there are N processors connected to the root processor. Let us denote

$$\lim_{N\to\infty} \alpha_j^N = \alpha_j^\infty. \quad (32)$$

Then

$$\alpha_j^\infty = \lim_{N\to\infty} f_j(N, \tau)/\sum_{j=0}^{N} f_j(N, \tau), \quad (33)$$

which can be written as

$$\alpha_j^\infty = \lim_{N\to\infty} \{f_{j-1}(N, \tau)/\sum_{j=0}^{N} f_j(N, \tau)\}\{f_j(N, \tau)/f_{j-1}(N, \tau)\}. \tag{34}$$

From (20) we again obtain the same continued fraction, which can be solved as

$$\lim_{N\to\infty} f_j(N, \tau)/f_{j-1}(N, \tau) = 1 + (1/2)\{\tau - \sqrt{\tau(\tau + 4)}\}. \tag{35}$$

Equation (34) can now be restated as

$$\alpha_j^\infty = \alpha_{j-1}^\infty\{1 + (\tau - \sqrt{\tau(\tau + 4)})/2\}, \tag{36}$$

where

$$\alpha_0^\infty = 1/S(0, \infty) = 1/\{(1/2)(1 + \sqrt{1 + (4/\tau)})\}. \tag{37}$$

Thus

$$\alpha_j^\infty = \{1 + (\tau - \sqrt{\tau(\tau + 4)})/2\}^j/\{(1 + \sqrt{1 + (4/\tau)})/2\}, \tag{38}$$

which can be further simplified to

$$\alpha_j^\infty = \{1 - 1/S(0, \infty)\}^j/S(0, \infty). \tag{39}$$

Equation (39) gives the optimal load distribution to the jth processor as $N \to \infty$. Thus, the load assigned to the jth processor asymptotically approaches the value α_j^∞ as the number of processors increases in the linear network.

3.3. *Analysis of Results*

The results obtained above in the asymptotic analysis are significant in many ways. The parameter τ defined in (5) has the following interpretation. It is the ratio of time delay in transmitting the entire load through one communication channel and the time taken to process the load in one processor. Figure 2 ($S(0, N)$ versus N for different values of τ) shows that the speed-up is higher for low values of τ but it quickly levels off after $N = 2$ or 3. Thus, the best speed-up can be achieved by adding at most two or three processors to the root processor regardless of the value of τ. The same information can be obtained from Fig. 3 ($S(N, N + n)$ versus n for a specific value of τ and various values of N). It shows that there is no significant speed-up by adding more processors when there are already $n = 2$ or 3 processors in the network.

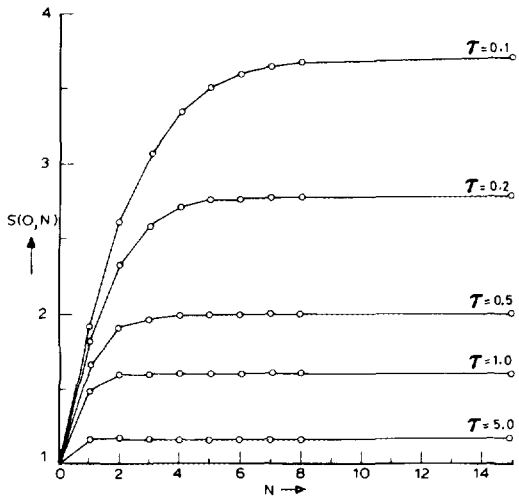

FIG. 2. $S(0, N)$ versus N for a linear network with front-end.

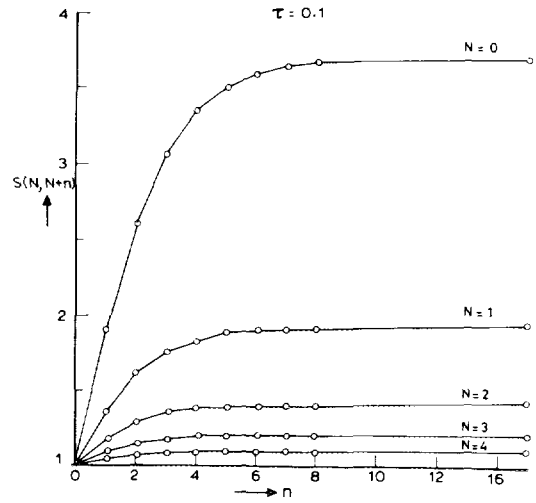

FIG. 3. $S(N, N + n)$ versus n for a linear network with front-end.

In view of the above, in an existing linear network having more than two or three processors attached to the root, it might be wiser to invest in obtaining better communication channels (i.e., lowering the value of z, and hence τ) rather than investing in more processors. Letting

$$\tau = 1/2^p, \tag{40}$$

we can obtain a relationship between $S(0, N)$ and p, which would show the variation in speed-up achieved, by adding N processors to the root processor, with improved communication channel performance. This can be obtained recursively using (24):

$$S(0, N) = \{1 + (1 + 1/2^p)(S(0, N - 1))\}/\{1 + (1/2^p)(S(0, N - 1))\}. \tag{41}$$

Here $S(0, 0) = 1$. It can be easily seen that

$$\lim_{p\to\infty} S(0, N) = N + 1 \tag{42}$$

From (23), we have

$$S(0, \infty) = (1/2)\{1 + \sqrt{1 + 2^{p+2}}\}. \tag{43}$$

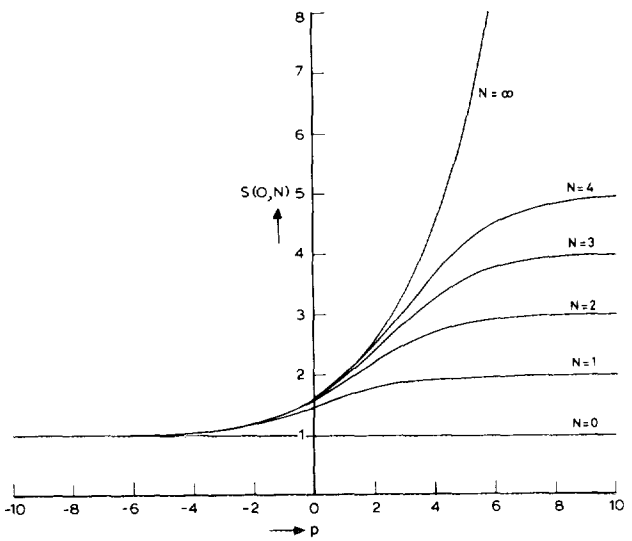

FIG. 4. $S(0, N)$ versus p for a linear network with front-end.

Thus as $p \to \infty$, $S(0, \infty) \to \infty$. For high values of $p > 0$, we can show that

$$S(0, \infty) \approx 2^{p/2} = 1/\sqrt{\tau}. \tag{44}$$

The above equation implies that for large values of N and p, when the communication delay is halved the speed-up achieved increases by a factor of approximately $\sqrt{2}$. Figure 4 shows the variation in $S(0, N)$ with p for various values of N. The load distribution to individual processors as $N \to \infty$ can be found from (37) and (39). For $\tau = 0.25$, $\alpha_0^\infty = 39.6\%$, $\alpha_1^\infty = 23.8\%$, $\alpha_2^\infty = 14.5\%$, $\alpha_3^\infty = 8.84\%$, and $\alpha_4^\infty = 5.38\%$. This accounts for more than 90% of the load. For higher values of τ, an even lesser number of processors share a large portion of the load. Thus, when there is a large number of processors, the processors after the first few are not utilized efficiently.

4. LINEAR NETWORK WITHOUT FRONT-END

In this section, we shall consider a linear network in which the processors are not equipped with front-ends. Here, too, we assume that all communication channels have the same speed and all processors have the same speed.

4.1. Basic Model and Closed-Form Expressions

Let the linear network with N processors connected to the root processor be as shown in Fig. 5, which also

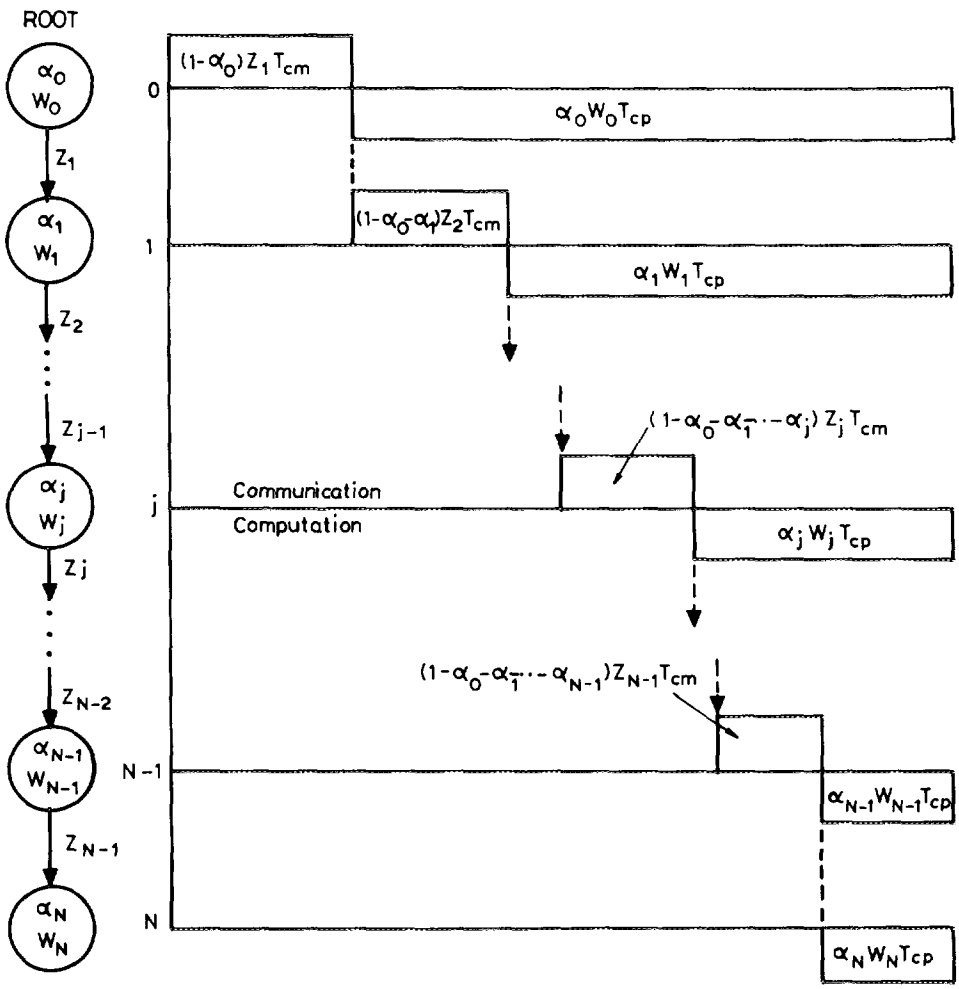

FIG. 5. Timing diagram for a linear network without front-end.

shows the timing diagram of the load distribution and the computational process. However, in this model, we have to make the additional assumption that

$$\tau < 1, \tag{45}$$

since, otherwise, a processor need not send the load to its immediate successor, as the load to be sent on can be processed at this processor itself in less time. The load distribution equations are as follows:

$$\alpha_i = (1 - \alpha_0 - \alpha_1 - \cdots - \alpha_{i+1})\tau + \alpha_{i+1}, \quad i = 0, 1, \ldots, N-1. \tag{46}$$

The normalizing equation is the same as in (3). As before, (46) along with the normalizing condition can be solved in closed-form by expressing all α_i, $i = 0, 1, \ldots, N-1$, in terms of α_N and substituting in (3). This operation yields [9]

$$\alpha_j = g_n(N, \tau)/\sum_{j=0}^{N} g_j(N, \tau), \quad j = 0, 1, \ldots, N, \tag{47}$$

where

$$g_j(N, \tau) = d(j, 0)\tau^0 + d(j, 1)\tau^1 + \cdots + d(j, L)\tau^L, \tag{48}$$

with

$$L = (N - j) \text{ div } 2 \tag{49}$$

and

$$d(j, i) = \{(N - j)!\}/\{(2i)!\,(N - j - 2i)!\}. \tag{50}$$

Equations (47)–(50) can be used to obtain the optimal load distribution among the $(N + 1)$ processors in a given linear network of distributed processors without front-ends. However, we shall not make explicit use of this closed-form expression in the subsequent analysis. The time taken to complete a computational task assigned to a linear network of N processors attached to the root processor is given by

$$T_N = wT_{cp}\{\tau + (1 - \tau)g_0(N, \tau)/\sum_{j=0}^{N} g_j(N, \tau)\}. \tag{51}$$

Here, $T_0 = wT_{cp}$ is the time taken by the root processor alone to process the entire load. Here, too, the speed-up achieved by connecting n additional processors is given by $S(N, N + n)$, and satisfies (12). However,

$$S(0, N) = 1/\{\tau + (1 - \tau)g_0(N, \tau)/\sum_{j=0}^{N} g_j(N, \tau)\}. \tag{52}$$

Using this, one may also derive $P(N, N + n)$ in a similar way through (15).

4.2. Asymptotic Analysis

We shall examine the behavior of $S(N, N + n)$ and $P(N, N + n)$ as $n \to \infty$. Again, consider a linear network with m processors connected to the root. Substituting $i = 0$ in (46), we get

$$\alpha_0 = (1 - \alpha_0 - \alpha_1)\tau + \alpha_1. \tag{53}$$

Substituting (47), we obtain

$$\left\{\sum_{j=0}^{m} g_j(m, \tau)/g_0(m, \tau)\right\} = 1 + (1/\tau) - \{(1/\tau - 1)g_1(m, \tau)/g_0(m, \tau)\}. \tag{54}$$

Now, substituting $i = k$ and $k + 1$ in (45) and performing some simple algebraic manipulations, we obtain the following recursive relationship:

$$\{g_{k+1}(m, \tau)/g_k(m, \tau)\} = 1/\{2 - (1 - \tau)g_{k+2}(m, \tau)/g_{k+1}(m, \tau)\}. \tag{55}$$

Thus

$$\lim_{m\to\infty} \{g_1(m, \tau)/g_0(m, \tau)\} = 1/\{2 - (1 - \tau)/\{2 - (1 - \tau)/\{2\ldots, \tag{56}$$

which is a continued fraction [8] and can be easily solved as

$$\lim_{m\to\infty} \{g_1(m, \tau)/g_0(m, \tau)\} = 1/(1 + \sqrt{\tau}), \tag{57}$$

from which

$$\lim_{m\to\infty} \left\{\sum_{j=0}^{m} g_j(m, \tau)/g_0(m, \tau)\right\} = 1 + 1/\sqrt{\tau}. \tag{58}$$

Therefore, from (52),

$$\lim_{m\to\infty} S(0, m) = S(0, \infty) = 1/\sqrt{\tau}. \tag{59}$$

The variation in $S(0, m)$ with m is obtained as

$$S(0, m) = \{1 + S(0, m - 1)\}/\{1 + \tau S(0, m - 1)\}, \tag{60}$$

with $S(0, 0) = 1$. Therefore

$$\lim_{n\to\infty} S(N, N + n) = (1/\sqrt{\tau})/S(0, N), \tag{61}$$

where $S(0, N)$ can be calculated using (60). We also obtain

$$P(0, \infty) = 1 - \sqrt{\tau}. \tag{62}$$

Using (60), we can derive the following recursive relationship:

$$P(0, m) = (1 - \tau)/\{2 - P(0, m - 1)\}. \tag{63}$$

If α_j^N denotes the computational load assigned to the jth processor when there are N processors connected to the root, then

$$\alpha_j^\infty = \lim_{N\to\infty} \alpha_j^N = \lim_{N\to\infty} g_j(N, \tau)/\sum_{j=0}^{N} g_j(N, \tau), \tag{64}$$

which can be written as

$$\alpha_j^\infty = \lim_{N\to\infty} \{g_{j-1}(N, \tau)/\sum_{j=0}^{N} g_j(N, \tau)\}\{g_j(N, \tau)/g_{j-1}(N, \tau)\}. \tag{65}$$

However, from (57),

$$\lim_{N\to\infty} \{g_j(N, \tau)/g_{j-1}(N, \tau)\} = 1/(1 + \sqrt{\tau}), \tag{66}$$

which implies that

$$\alpha_j^\infty = \alpha_{j-1}^\infty/(1 + \sqrt{\tau}), \tag{67}$$

where, from (58),

$$\alpha_0^\infty = \sqrt{\tau}/(\sqrt{\tau} + 1). \tag{68}$$

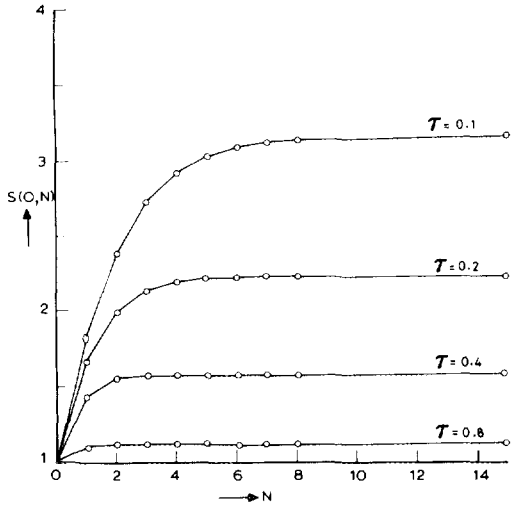

FIG. 6. $S(0, N)$ versus N for a linear network without front-end.

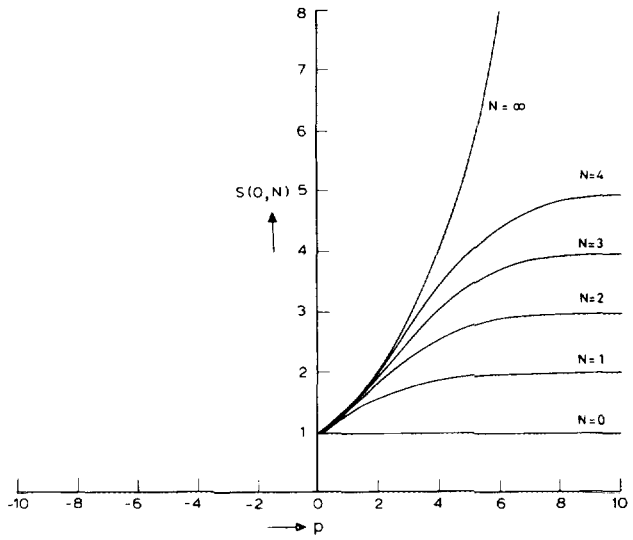

FIG. 7. $S(0, N)$ versus p for a linear network without front-end.

Thus

$$\alpha_j^\infty = \sqrt{\tau}/(\sqrt{\tau} + 1)^{j+1}. \tag{69}$$

The above relation gives the optimal load distribution to the jth processor as $N \to \infty$. Thus, the load assigned to the jth processor asymptotically approaches the value a_j^∞ as the number of processors increases in a linear network.

4.3. Analysis of Results

Figure 6 ($S(0, N)$ versus N for different values of τ) shows that the speed-up is higher for low values of τ but is almost insignificant for higher values. As expected, the speed-up is less, for a given value of τ, than in the case when the processors are equipped with front-end. Here too, there is no significant benefit in adding more than two or three processors to the root.

To study the effect of decreasing τ, let τ be defined as in (40). Then, from (60),

$$\lim_{p\to\infty} S(0, N) = N + 1, \tag{70}$$

which is the same as in the case of processors with front-end. In fact, we can see that when τ is small the adverse effect of the absence of front-end is much less than when τ is high. From (59),

$$S(0, \infty) = 2^{p/2} = 1/\sqrt{\tau}. \tag{71}$$

Hence, when the communication delay is halved the speed-up approximately increases by a factor of $\sqrt{2}$. This is true for all value of $\tau < 1$, and for reasonably large values of N (for example, $N \geq 3$ when $\tau = 0.4$ or $N \geq 5$ when $\tau = 0.2$). Figure 7 shows the variation in $S(0, N)$ with p for various values of N. Values of $p > 0$ are con-

sidered since $\tau < 1$. For low values of p (i.e., high values of τ) the speed-up achieved is substantially less compared to the case with front-end, whereas for high values of p the speed-up in both cases are almost same.

5. SINGLE-LEVEL TREE NETWORK WITH FRONT-END

In this section, we consider a single-level tree network in which the root processor, which distributes the load, is equipped with a front-end.

5.1. Basic Model and Closed-Form Expressions

Let the single-level tree network with N processors connected to the root be as given in Fig. 8, which also shows the timing diagram of the load distribution and the computational process. The load distribution equations are as follows:

$$\alpha_i = (1 + \tau)\alpha_{i+1}, \quad i = 0, 1, \ldots, N - 1. \tag{72}$$

The normalizing equation is

$$\alpha_0 + \alpha_1 + \cdots + \alpha_N = 1. \tag{73}$$

These equations are easily solved in closed-form by expressing all α_i, $i = 0, 1, \ldots, N - 1$, in terms of α_N and substituting in (73). As shown by Bharadwaj *et al.* [4], this yields

$$\alpha_j = \{\tau(1 + \tau)^{N-j}\}/\{(1 + \tau)^{N+1} - 1\}, \quad j = 0, 1, \ldots, N. \tag{74}$$

Equation (74) can be used to obtain the optimal load distribution among the $(N + 1)$ processors. The time taken to complete a computational task assigned to a single-level tree network of N processors attached to the root processor is given by

$$T_N = wT_{cp}\{\tau(1 + \tau)^N\}/\{(1 + \tau)^{N+1} - 1\}. \tag{75}$$

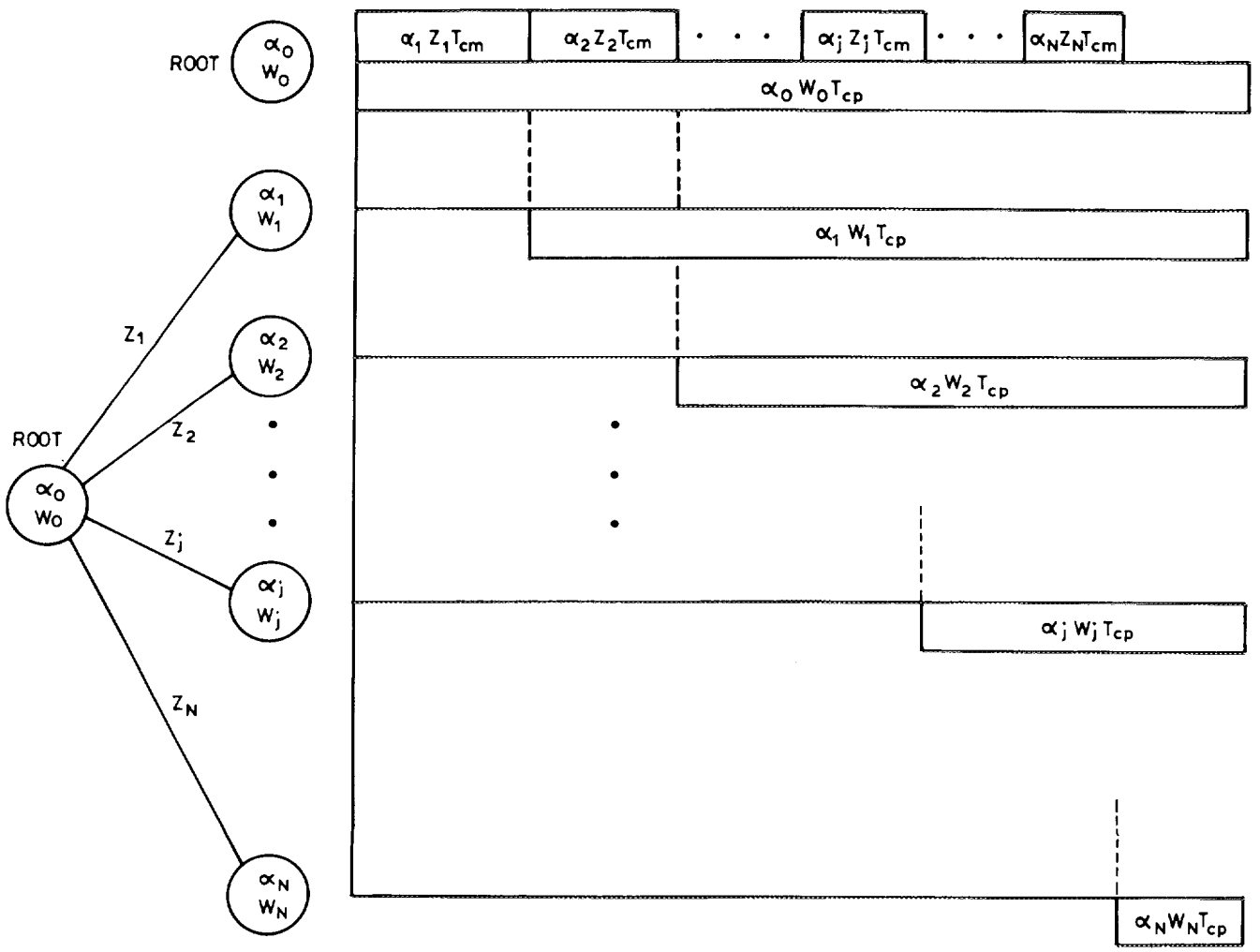

FIG. 8. Timing diagram for a single-level tree network with front-end.

Here, $T_0 = wT_{cp}$ is the time taken by the root processor alone to process the entire processing load.

The speed-up achieved by connecting n additional processors to the existing N processors is given by $S(N, N + n)$, which satisfies (12). Here

$$S(0, N) = \{(1 + \tau)^{N+1} - 1\}/\{\tau(1 + \tau)^N\}, \quad (76)$$

from which

$$S(N, N + n) = [\{(1 + \tau)^{N+n+1} - 1\}/\{(1 + \tau)^{N+1} - 1\}] (1 + \tau)^n. \quad (77)$$

The factor $P(N, N + n)$ can be defined through (15).

5.2. Asymptotic Analysis

Here

$$S(0, \infty) = \lim_{N \to \infty} S(0, N) = 1 + 1/\tau. \quad (78)$$

Using (12), we have

$$\lim_{n \to \infty} S(N, N + n) = (1 + \tau)^{N+1}/\{(1 + \tau)^{N+1} - 1\}. \quad (79)$$

The variation of $S(0, N)$ with N can also be obtained through the recursive relationship

$$S(0, N) = \{1/(1 + \tau)^N\} + S(0, N - 1), \quad (80)$$

with $S(0, 0) = 1$. We also obtain

$$P(0, \infty) = 1/(1 + \tau), \quad (81)$$

and, from (15),

$$P(0, N) = \{(1 + \tau)^N - 1\}/\{(1 + \tau)^{N+1} - 1\}. \quad (82)$$

If α_j^N denotes the computational load assigned to the jth processor when there are N processors connected to the root, then, from (74),

$$\alpha_j^\infty = \lim_{N \to \infty} \alpha_j^N = \tau/(1 + \tau)^{j+1}, \quad j = 0, 1, \ldots, N. \quad (83)$$

This relation gives the optimal load distribution to the jth processor as $N \to \infty$. Thus, the load assigned to the jth processor asymptotically approaches the value α_j^∞ as the number of processors increase in the single-level tree network.

5.3. Analysis of Results

Figure 9 ($S(0, N)$ vs. N for different values of τ) shows that for low values of τ the speed-up saturates after a larger number of processors than in the case of linear network. It implies that when the communication speed is high, connecting additional processors in a single-level tree network is more beneficial than in a linear network. However, for high values of τ this is not true. We also note that the speed-up saturation level is higher for a tree network than for a linear network.

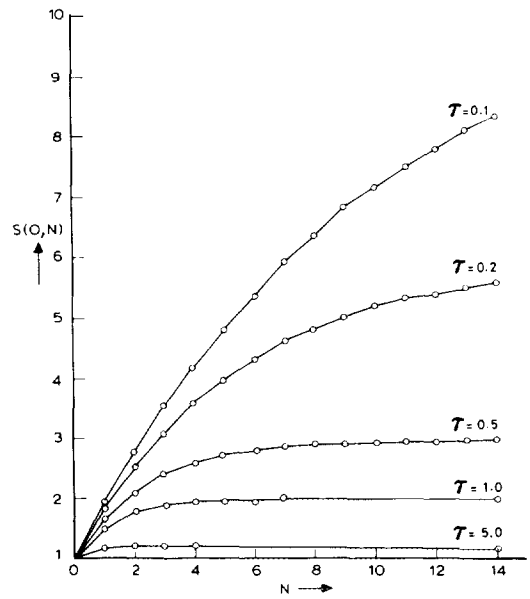

FIG. 9. $S(0, N)$ versus N for a single-level tree network with front-end.

To study the effect of decreasing τ on $S(0, N)$, let τ be defined as given in (40). Then

$$\lim_{p \to \infty} S(0, N) = \lim_{\tau \to \infty} \{(1 + \tau)^{N+1} - 1\}/\{\tau(1 + \tau)^N\} = N + 1. \quad (84)$$

From (78),

$$S(0, \infty) \approx 2^p, \quad (85)$$

for large values of p. This implies that the speed-up increases almost by a factor of 2 when the communication delay is halved. This shows better performance than in case of linear networks. Figure 10 shows the variation $S(0, N)$ with p for various values of N. We note that for low values of p (i.e., high values of τ or slow communication channels) the performance of the tree network is better than that of the linear network, whereas for high values of p the performance of both networks is almost the same.

6. SINGLE-LEVEL TREE NETWORK WITHOUT FRONT-END

In this section, we consider a single-level tree network in which the root processor is not equipped with a front-end.

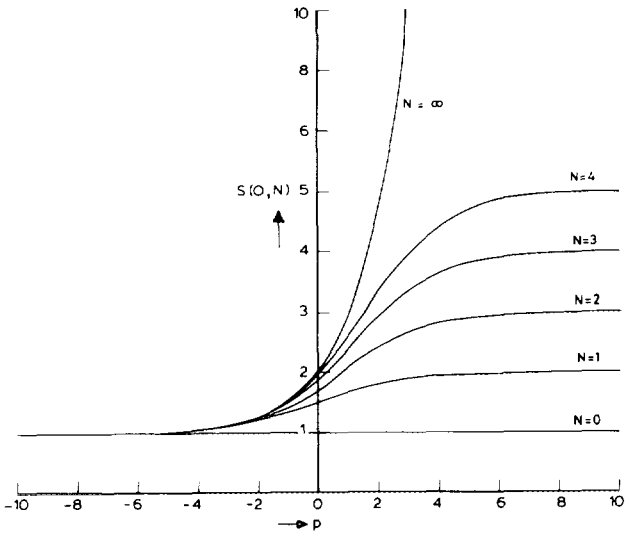

FIG. 10. $S(0, N)$ versus p for a single-level tree network with front-end.

6.1. Basic Model and Closed-Form Expressions

Let the single-level tree network with N processors connected to the root be as shown as in Fig. 11, which also shows the timing diagram of the load distribution and the computational process. Here also we assume that $\tau <$ 1 for reasons given in Section 4.1. The load distribution equations are as follows:

$$\alpha_i = (1 + \tau)\alpha_{i+1}, \quad i = 1, 2, \ldots, N - 1, \tag{86}$$

$$\alpha_N = \alpha_0. \tag{87}$$

The normalizing equation is

$$\alpha_0 + \alpha_1 + \cdots + \alpha_N = 1. \tag{88}$$

These equations can also be solved in closed-form by expressing all α_i, $i = 0, 1, \ldots, N - 1$, in terms of α_n and then using Eq. (88). This yields [4]

$$\alpha_0 = \alpha_N \tag{89}$$

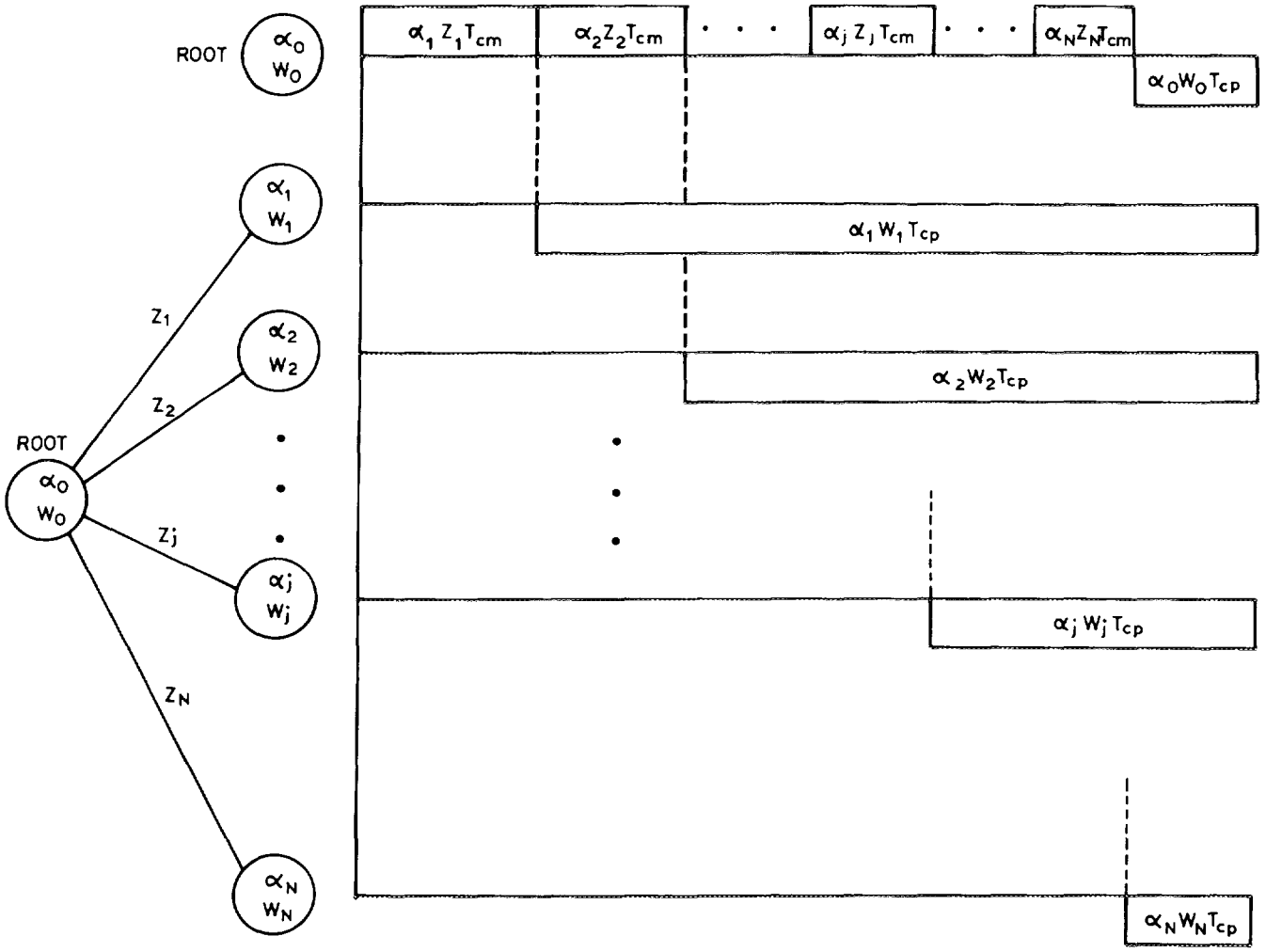

FIG. 11. Timing diagram for a single-level tree network without front-end.

$$\alpha_j = \{\tau(1 + \tau)^{N-j}\}/\{(1 + \tau)^N + \tau - 1\},$$
$$j = 1, 2, \ldots, N. \quad (90)$$

Thus the time taken to complete a computational task is

$$T_N = wT_{cp}\{\tau(1 + \tau)^N\}/\{(1 + \tau)^N + \tau - 1\}. \quad (91)$$

Here $T_0 = wT_{cp}$ is the time taken by the root processor alone to process the entire load.

The speed-up achieved by connecting n additional processors to the existing N processors is given by $S(N, N + n)$, which satisfies (12). Here

$$S(0, N) = \{(1 + \tau)^N + \tau - 1\}/\{\tau(1 + \tau)^N\}, \quad (92)$$

from which

$$S(N, N + n) =$$
$$[\{(1 + \tau)^{N+n} + \tau - 1\}/\{(1 + \tau)^N + \tau - 1\}](1 + \tau)^n. \quad (93)$$

The factor $P(N, N + n)$ can be defined through (15).

6.2. Asymptotic Analysis

We have

$$S(0, \infty) = \lim_{N\to\infty} S(0, N) = 1/\tau. \quad (94)$$

Using (12), we have

$$\lim_{n\to\infty} S(N, N + n) = (1 + \tau)^N/\{(1 + \tau)^N + \tau - 1\}. \quad (95)$$

The variation of $S(0, N)$ with N can also be obtained through the recursive relationships

$$S(0, N) = \{(1 - \tau)/(1 + \tau)\} + S(0, N - 1), \quad (96)$$

with $S(0, 0) = 1$. We also obtain, from (15),

$$P(0, N) = (1 - \tau)\{(1 + \tau)^N - 1\}/\{(1 + \tau)^N + \tau - 1\} \quad (97)$$

and

$$P(0, \infty) = 1 - \tau. \quad (98)$$

Also, from (90),

$$\alpha_j^\infty = \lim_{N\to\infty} \alpha_j^N = \tau/(1 + \tau)^j, \quad j = 1, 2, \ldots, N. \quad (99)$$

$$\alpha_0^\infty = \alpha_N^\infty. \quad (100)$$

Hence the load assigned to the jth processor asymptotically approaches the value α_j^∞ as the number of processors increases in the single-level tree network.

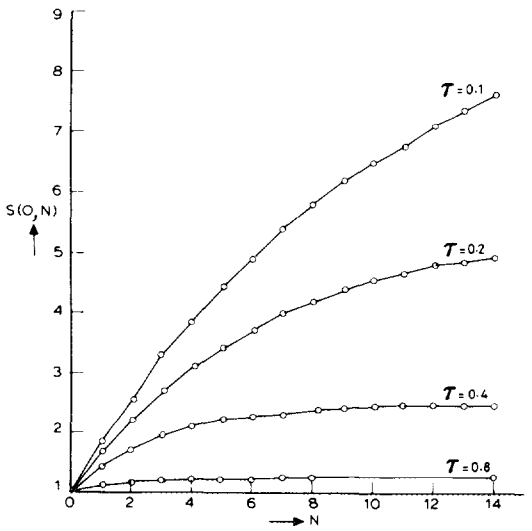

FIG. 12. $S(0, N)$ versus N for a single-level tree network without front-end.

6.3. Analysis of Results

Figure 12 ($S(0, N)$ versus N for different values of τ) shows a similar phenomena discussed earlier. To study the effect of decreasing τ on $S(0, N)$, let τ be defined as in (40). Then

$$\lim_{p\to\infty} S(0, N) = \lim_{\tau\to\infty} \{(1 + \tau)^N + \tau - 1\}/\{\tau(1 + \tau)^N\}$$
$$= N + 1. \quad (101)$$

From (94),

$$S(0, \infty) = 2^p. \quad (102)$$

Hence, the speed-up increases almost by a factor 2 when the communication delay is halved, in cases where N is reasonably large. Figure 13 shows the variation in $S(0, N)$ with p for various values of N.

7. CONCLUSIONS

In this paper, an asymptotic analysis was carried out to obtain the limit on performance enhancement by using additional processors in distributed linear and single-level tree networks with communication delays. As the sequential portion of an algorithm limits the speed-up in parallel computing, it is found that communication delays have a similar limiting effect on the performance enhancement in a distributed computing network. Closed-form expressions are obtained on the limiting effect of communication delay for the speed-up and fractional savings in time. These closed-form expressions are useful in obtaining analytical results which would otherwise require extensive computation. This study also shows that in both linear and single-level tree architectures signifi-

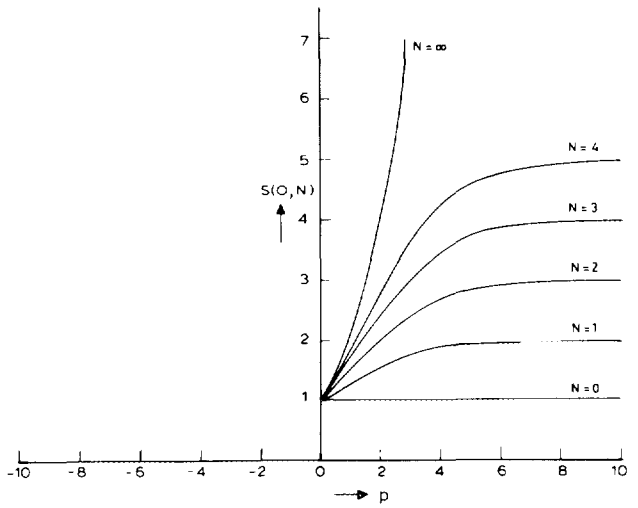

FIG. 13. $S(0, N)$ versus p for a single-level tree network without front-end.

cant performance enhancement is achieved for only the first few processors. Adding more processors leads to underutilization of the additional processors without causing any significant improvement in overall performance.

Though this paper carries out the analysis for linear and single-level tree architectures only, it seems possible to extend these results to other architectures, also. A drawback of the model assumed here is that the modeling of the communication channel delay is somewhat simplified. A more realistic model may give a quantitatively different result, but the qualitative nature of the performance curves is not expected to be much different.

ACKNOWLEDGMENT

The authors appreciate the discussions with Mr. V. Bharadwaj on some aspects of the problems addressed in this paper.

REFERENCES

1. Bataineh, S., and Robertazzi, T. G. Bus-oriented load sharing for a network of sensor driven processors. *IEEE Trans. Systems Man Cybernet.* **21,** 5 (Sep.–Oct. 1991), 1202–1205.
2. Bataineh, S., and Robertazzi, T. G. Ultimate performance limits for networks of load sharing processors. *Proceedings of the 1992 Conference on Information and Systems,* Princeton University, Princeton NJ, Mar. 1992, pp. 794–799.
3. Bertsekas, D. P., and Tsitsiklis, J. N. *Parallel and Distributed Computation: Numerical Methods.* Prentice–Hall, Englewood Cliffs, NJ, 1989.
4. Bharadwaj, V., Ghose, D., and Mani, V. Optimal sequencing and arrangement in distributed single level tree networks with communication delays. *IEEE Trans. Parallel Distribut. Systems,* **5,** (1994).
5. Bokhari, S. H. *Assignment Problems in Parallel and Distributed Computing.* Kluwer, Boston, 1987.
6. Cheng, Y. C., and Robertazzi, T. G. Distributed computation with communication delay. *IEEE Trans. Aerospace Electron. Systems* **24,** 6 (Nov. 1988), 700–712.
7. Cheng, Y. C., and Robertazzi, T. G. Distributed computation for tree network with communication delay. *IEEE Trans. Aerospace Electron. Systems* **26,** 3 (May 1990), 511–516.
8. Graham, R. L., Knuth, D. E., and Patashnik, O. *Concrete Mathematics.* Addison–Wesley, Reading, MA, 1989.
9. Mani, V., and Ghose, D. Distributed computation in linear network: Closed-form solutions. *IEEE Trans. Aerospace Electron. Systems,* **30,** 2 (April 1994), 471–483.
10. Mirchandaney, R., Towsley, D., and Stankovic, J. A. Analysis of the effects of delays on load sharing. *IEEE Trans. Comput.* **38** (1989), 1513–1525.
11. Shivaratri, N. G., Krueger, P., and Singhal, M. Load distribution for locally distributed systems. *IEEE Comput. Mag.* **25** (1992), 33–44.
12. Quinn, M. J. *Designing Efficient Algorithms for Parallel Computers,* McGraw–Hill, New York, 1987.

Received May 20, 1992; revised September 13, 1993; accepted September 17, 1993

D. GHOSE received the B.Sc. (Engg) degree in electrical engineering from the Regional Engineering College, Rourkela, in 1982 and the M.E. and Ph.D. degrees, also in electrical engineering, from the Indian Institute of Science, Bangalore, India in 1984 and 1990, respectively. From 1984 to 1987, he worked as a scientific officer in the Joint Advanced Technology Programme at the Indian Institute of Science. He is now an assistant professor in the Department of Aerospace Engineering, Indian Institute of Science. His research interests include distributed computing, guidance and control of aerospace vehicles, game theory, and mathematical economics.

V. MANI received the B.E. degree in civil engineering from Madurai University in 1974, the M. Tech. degree in aeronautical engineering from the Indian Institute of Technology, Madras, India in 1976, and the Ph.D. degree in engineering from the Indian Institute of Science, Bangalore, India in 1986. From 1986 to 1988, he was a research associate in the School of Computer Science at the University of Windsor, Windsor, Ontario, Canada, and from 1989 to 1990 in the Department of Aerospace Engineering at the Indian Institute of Science. Since 1990 he has been an assistant professor in the Department of Aerospace Engineering, Indian Institute of Science. His research interests include distributed computing, queueing networks, reliability, neural computing, and mathematical modeling.