Multi-installment Load Distribution in Tree Networks With Delays

V. BHARADWAJ
D. GHOSE
V. MANI
Indian Institute of Science
India

This paper presents a new strategy for load distribution in a single-level tree network equipped with or without front-ends. The load is distributed in more than one installment in an optimal manner to minimize the processing time. This is a deviation and an improvement over earlier studies in which the load distribution is done in only one installment. Recursive equations for the general case, and their closed-form solutions for a special case in which the network has identical processors and identical links, are derived. An asymptotic analysis of the network performance with respect to the number of processors and the number of installments is carried out. Discussions of the results in terms of some practical issues like the tradeoff relationship between the number of processors and the number of installments are also presented.
[13] presented an asymptotic analysis of linear and single-level tree networks and obtained bounds on performance enhancement achieved by connecting additional number of processors in an existing network. Robertazzi [15], and Batanieh and Robertazzi [16], also obtained some performance limits, through a different approach using the concept of processor equivalence.

All the above studies implicitly assume that a processor starts computing only after it has received all the load assigned to it. This gives rise to considerable idle time for almost all the processors because of the delay involved in communicating load from one processor to another. We relax the above-mentioned assumption and present a new strategy of load distribution in a single-level tree network, in which the load assigned to a processor is sent in more than one installment. Using this strategy the idle time of the processors can be reduced to a great extent, thereby circumventing the effect of communication delays. The performance obtained by this strategy is shown to be better in many ways than the earlier approaches. We also provide an asymptotic analysis of this new strategy.

All the above analysis is done for two cases: 1) when the processors are equipped with front-ends and 2) when the processors are not equipped with front-ends. In 1) a processor can be engaged in computing a given load fraction while its front-end is simultaneously busy receiving the next installment. On the other hand, in 2) the computation and communication cannot be done simultaneously. A processor can be engaged doing only one of these at any given instant in time.

The organization of the paper is as follows. Section II provides the motivation for the new approach and gives some important definitions and states some necessary conditions for optimal load distribution. Section III develops general recursive equations for load distribution and Section IV provides closed-form solutions for the load distribution. Section V and VI gives asymptotic analysis, results, and discussions. Section VII concludes this work.

II. MOTIVATION AND PRELIMINARY RESULTS

In this section, using a simple example, we show that the processing time can be decreased by distributing the load in two installments rather than in one. Further we formally define some important concepts and state some necessary conditions, which forms the basis for optimal load distribution in installments.

A. Motivation for Installment Technique

Consider a single-level tree network, which consists of one root processor and one child processor. The total processing load has to be shared in an optimal manner such that the processing time is minimum. In the timing diagram for load distribution given in [7, 12], it can be seen that a child processor has to wait for a certain amount of time before it actually starts computing. Hence, if by some means this waiting time can be reduced, so that the child processor can start computing at an earlier point in time, then we can further reduce the processing time. One way of reducing the waiting time is by sending the processing load in more than one installment. The following simple example illustrates this method, in the case where the processors are equipped with front-ends. Fig. 1 shows the case when the load is sent in one installment and Fig. 2 shows the case when the load is sent in two installments. From Figs. 1 and 2, we can see that the processing time is reduced by 9.6% by sending the load in two installments. Obviously, the processing time can be reduced further by increasing the number of installments.

Fig. 1. Load distribution in one installment.

Fig. 2. Load distribution in two installments.
B. Definitions and Some Results

Now we consider the general case of a single-level tree network consisting of one root processor \( p_0 \) and \( m \) child processors \((p_1, \ldots, p_m)\), connected to \( p_0 \). The root processor distributes the load assigned to each child processor in \( n \) installments.

In practice, a processing load cannot be divided into infinitesimal small installments. The extent to which the load can be divided into installments is governed by a parameter called the “divisibility factor” defined below.

**Divisibility Factor:** Let \( \alpha \) be the smallest fraction of the total load that can be assigned to a processor in any installment such that \( \alpha \in (0, 1] \). Then the divisibility factor \( \delta \) is defined as,

\[
\delta = (1/\alpha).
\]  

when \( \alpha \) is arbitrarily small, the load is said to be infinitely divisible.

In all the earlier studies [6–16] this factor was implicitly assumed to be infinite.

**Processing Time:** The processing time, denoted by \( T(m, n) \), is defined as

\[
T(m, n) = \max(T_0, \ldots, T_m)
\]  

where \( T_k \) is the time difference between the instant at which the \( k \)th processor stops processing and the instant at which the root processor initiates the process.

**Optimal Load Distribution:** This is defined as that load distribution for which the processing time \( T(m, n) \) is minimum.

The other parameters used in this work are \( z \) and \( w \) which are inversely proportional to the speed of the links and processors, respectively. The parameters \( T_{cm} \) and \( T_{cp} \) denote the communication time and computation time of a standard link and processor, respectively, for the entire load. Note that for a standard link and for a standard processor \( z = 1 \) and \( w = 1 \).

The load distribution in installments is assumed to follow the rules given below.

1) The root processor \( p_0 \) keeps a fraction of the total processing load for itself to compute and distributes the rest to the child processors.

2) The root processor distributes loads to its children in installments following a given fixed sequence. (When the processors are equipped with front-ends, this is actually done by the front-ends).

3) The root processor assigns load only to those link-processor pairs \((l_k, p_k)\) which satisfy

\[
z_kT_{cm} < z_{k+1}T_{cm} + w_{k+1}T_{cp}
\]  

where \((l_k, p_k)\) and \((l_{k+1}, p_{k+1})\) are two adjacent link-processor pairs in the sequence of load distribution followed in 2). In addition, in the case where processors are not equipped with front-ends, the root processor assigns load only to those processors \( p_k \) which satisfy

\[
w_kT_{cp} > z_kT_{cm}.
\]

4) The front-end of the root processor (or the root processor itself when there is no front-end) is continuously engaged in distributing the load until all the installments have been communicated.

5) When the processors are equipped with front-ends, they all must be engaged continuously in computation once they start computing the load fractions from the first installment till the end of the last installment. On the other hand, when the processors are not equipped with front-ends, all the processors must be engaged continuously in computing a given installment once it is received.

6) All the processors stop computing at the same time.

7) A processor starts computing a given installment, the moment its front-end finishes receiving that installment.

Rule 3 needs some explanation. If condition (3) is violated then the time taken by the front-end of \( p_0 \) to communicate load fraction to \( p_k \) through \( l_k \) is more than the time taken to communicate the same load through \( l_{k+1} \) and process it at \( p_{k+1} \). Hence it is logical to send the load to \( p_{k+1} \) rather than to \( p_k \). Similarly, if condition (4) is violated, then the time taken for communicating a load fraction to \( p_k \) through \( l_k \) is more than the time taken for processing the load at \( p_0 \) itself. Hence it is logical that the root does not assign any load to \( p_k \) through \( l_k \). The conditions (3) and (4) are necessary for optimality [7, 8] but are not sufficient. In [14], for a single-level tree network consisting of nonidentical processors and nonidentical links, both necessary and sufficient conditions for load distribution has been presented. In the present study we assume that all the processors and links are such that all processors are utilized in processing, i.e., the inefficient processor-link pairs have already been deleted. Rules 4), 6), and 7) have been implicitly assumed in earlier studies [6–16].

III. MATHEMATICAL MODEL AND LOAD DISTRIBUTION EQUATIONS

A. The Model

The architecture consists of one root processor and \( m \) child processors as shown in Fig. 3. The processors are designated as \( p_m, p_{m-1}, \ldots, p_1 \) and the communication links as \( l_m, l_{m-1}, \ldots, l_1 \). This tree configuration can be represented as an ordered set as follows:

\[
T(p_0) = \{(l_m, p_m), (l_{m-1}, p_{m-1}), \ldots, (l_1, p_1), \ldots, (l_1, p_1)\}
\]  


where \((l_i, p_i)\) represents the \(i\)th processor \(p_i\) is connected to \(p_0\) via link \(l_i\). The load distribution starts sequentially from left end of the tree to the right end of the tree which is also the order mentioned in (5). Note that here the links and processors are renumbered in a reverse order compared with the numbering followed in the previous section. Further, the installments are also numbered in a reverse order. This notation is adopted for ease of obtaining the load distribution equations and their solution (either computationally or analytically) for a general case of arbitrary number of processors and installments. Thus, for example, in this notation, in a four processor system (1 root and 3 child processors, i.e., \(m = 3\)) with three installments \((n = 3)\) we have the load distribution as follows: \(a_0\) to the root processor and \(a_{3,1}, a_{3,2}, a_{1,3}\) in the first installment, \(a_{3,1}, a_{2,3}, a_{1,2}\) in the second installment, and \(a_{3,1}, a_{2,1}, a_{1,1}\) in the third installment, to \(p_3, p_2, p_1\), respectively, from the left end to the right end of the tree. These fractions satisfy rule 5), since (7)-(10) are obtained by assuming rules 1)-7).

### B. Load Distribution with Front-End

Using the rules mentioned in Section II, for a system of \((m + 1)\) processors with load distributed in \(n\) installments, the timing diagram is as shown in Fig. 4.

From this diagram we can write the following recursive equations:

\[
\alpha_{i,j} w_i T_{xp} = \alpha_{i-1,j} (z_{i-1} T_{cm} + w_{i-1} T_{cp}),
\]

\(i = 2, 3, \ldots, m\) (6)

Denoting \(\sigma = T_{cm}/T_{cp}\), from (6) we have,

\[
\alpha_{i,i} = \alpha_{1,i} \left\{ \prod_{k=1}^{i-1} \left[ (w_k + z_k \sigma)/w_{k+1} \right] \right\}
\]

\(i = 1, \ldots, m; \quad j = 2, \ldots, n\), (7)

and the load on the root processor is

\[
\alpha_0 = \left\{ \sum_{i=1}^{m} \sum_{j=1}^{n} \alpha_{i,j} z_i \right\} \sigma/w_0 + \alpha_{1,1} w_1/w_0.
\]

The normalizing equation is given by,

\[
\alpha_0 + \alpha_{1,1} + \cdots + \alpha_{m,n} = 1.
\]

Note that in (8) the first summation in the right-hand side (RHS) vanishes when \(i = 1\). Thus, we have totally \((mn + 1)\) linear equations and \((mn + 1)\) unknowns. These equations can be solved recursively by initially assuming \(\alpha_{1,1} = 1\) and using (7) to obtain \(\alpha_{i,1}\) for \(i = 2, 3, \ldots, m\). The remaining \(\alpha_{i,j}\) can be computed.
Using (8). Then the value of $a_0$ is computed using (9). Using the normalizing equation (10) the exact value of $a_{1,1}$ and the remaining $a_{i,j}$ and $a_0$ can be computed.

Then the processing time can be computed as the processing time of the root processor ($a_0 w_0 T_{cp}$).

C. Load Distribution Without Front-End

When the processors are not equipped with front-end, in the case when $m = 1$, it can be verified that the performance is independent of the number of installments. Hence, the model considered is only valid for $m \geq 2$. The timing diagram is shown in Fig. 5, from which the following recursive equations can be obtained:

$$a_{i,1} w_i T_{cp} = a_{i-1,1} (w_{i-1} T_{cp} + z_{i-1} T_{cm}),$$

$$i = 2, \ldots, m.$$  

(11)

From (11), we have

$$a_{i,1} = a_{1,1} \left[ \prod_{k=1}^{i-1} \left( \frac{w_k + z_k \sigma}{w_{k+1}} \right) \right]$$

(12)

$$a_{i,j} = \left\{ \sum_{k=1}^{i-1} a_{k,j-1} z_k + \sum_{k=i+1}^{m} a_{k,i-1} z_k \right\} \sigma/w_i,$$

$$i = 1, \ldots, m; \quad j = 2, \ldots, n$$

(13)

and the load on the root processor is

$$a_0 = a_{1,1} w_1/w_0.$$  

(14)

The normalizing equation is given by (10). Note that, as in the previous case, the first summation in the RHS of (13) vanishes when $i = 1$ and the second summation vanishes when $i = m$.

Following the same procedure described in the earlier case these recursive equations can be used to compute the individual loads to processors in each installment and the processing time.

Since the closed-form solution for the processing time in a general case (with and without front-ends) is difficult to derive, in the next section we give closed-form solutions for a special case where all the link speeds are equal and all the processors speeds are also equal.

IV. CLOSED-FORM SOLUTIONS

In this section we derive closed-form solutions for the single-level tree network for a special case. When all the link speeds are the same, this architecture behaves like the bus architecture [9]. This means that $z_i = z$, $i = 1, 2, \ldots, m$. Furthermore, assume that all the processor speeds are equal, i.e., $w_i = w$, $i = 1, 2, \ldots, m$. This assumption implies that all the processors in the network are identical and the communication links are also identical. We denote the factor

$$\tau = z \sigma/w.$$  

(15)

A. Single Level Tree With Front-End

In this case (7)-(9) can be rewritten as

$$a_{i,1} = a_{1,1} (1 + \tau)^{-1}, \quad i = 2, \ldots, m$$

(16)
\[
\alpha_{i,j} = \left( \sum_{k=1}^{i-1} \alpha_{k,j} + \sum_{k=i}^{m} \alpha_{k,j-1} \right) \tau_i, \quad i = 1, \ldots, m; \quad j = 2, \ldots, n \tag{17}
\]
\[
\alpha_0 = \left( \sum_{i=1}^{m} \sum_{j=1}^{n} \alpha_{i,j} \right) \tau + \alpha_{1,1} \tag{18}
\]

For further simplification we use the following notation
\[
\alpha_{i,j} = \alpha_{1,1} \chi_k, \quad i = 1, \ldots, m; \quad j = 1, \ldots, n \tag{19}
\]
where,
\[
k = (i - 1) + (j - 1)m. \tag{20}
\]

Then, from (16)-(18) we obtain
\[
X_k = (1 + \tau)^k, \quad k = 0, 1, \ldots, m - 1, \tag{21}
\]
\[
X_k = (X_{k-1} + X_{k-2} + \ldots + X_{k-m}) \tau, \quad k = m, m + 1, \ldots, mn - 1, \tag{22}
\]
\[
\alpha_0 = \alpha_{1,1} \tau (X + 1) \tag{23}
\]
where
\[
X = \sum_{i=0}^{mn-1} X_i \tag{24}
\]

Using normalizing equation (10) we have,
\[
\alpha_{1,1} = 1/\left[1 + (1 + \tau)X\right]. \tag{25}
\]

Thus, the processing time \( T(m, n) \) is obtained as
\[
T(m, n) = [(\tau X + 1)/{(1 + \tau)X + 1}]\mu T_{cp}. \tag{26}
\]

Hence, once the value of \( X \) is known, the processing time \( T(m, n) \) can be obtained. In order to obtain \( X \), we have to compute the values of \( X_i, i = 0, 1, \ldots, mn - 1. \) This can be done by using a generating functions approach. The generating function should have as its coefficients the values generated by the recursive equations (22) with initial conditions (21). Note that the index of the recursive equation, which provides the coefficients in the generating function, is not limited to \( mn - 1 \) but goes to infinity. However, for the purpose of computing \( T(m, n) \), we consider terms from \( X_0 \) and \( X_{mn-1} \) only. The generating function is given as
\[
G(s) = X_0 + X_1 s + \ldots + X_{mn-1} s^{mn-1}. \tag{27}
\]

Following the procedure given in [17],
\[
G(s) = \frac{(1 + s + \ldots + s^{m-1})}{\{1 - \tau s(1 + s + \ldots + s^{m-1})\}} \tag{28}
\]

This can be expanded into a power series in which the coefficients of \( s^i \) is \( X_i \). The expansion into power series can be done in two ways. The first method is by using the Rational Expansion Theorem stated in [17] and the other method is by using binomial expansion of the numerator and the denominator of (28). We show how both the methods lead to a closed-form solution.

1) Solution Using Rational Expansion Theorem:

If \( G(s) = P(s)/Q(s) \) and \( Q(s) \) has distinct roots as \( \frac{1}{\mu_1}, \frac{1}{\mu_2}, \ldots, \frac{1}{\mu_m} \), then we can express \( X_k \) as
\[
X_k = a_1(\mu_1)^k + a_2(\mu_2)^k + \ldots + a_m(\mu_m)^k, \quad k = 0, 1, \ldots, mn - 1. \tag{29}
\]

where,
\[
a_j = -\mu_j P(1/\mu_j)/Q'(1/\mu_j), \quad j = 1, \ldots, m \tag{30}
\]

where \( Q'(1/\mu_j) \) is the first-order derivative of \( Q(s) \) evaluated at \( (1/\mu_j) \). So, here the numerator \( P(s) \) and the denominator \( Q(s) \), from (28) are given as,
\[
P(s) = 1 + s + \ldots + s^{m-1} \tag{31}
\]
\[
Q(s) = 1 - \tau s P(s). \tag{32}
\]

From (29)-(32), we have,
\[
X_k = (1/\tau) \sum_{i=0}^{m-1} \frac{(\mu_i)^{k+i}(\mu_i-1)}{\mu_i \{\mu_i\}^{m-1} - m(\mu_i-1)} \tag{33}
\]

The above equation (33) shows that \( X_k \) can be expressed in terms of the roots of \( Q(s) \) alone. Substituting in (24) we have,
\[
X = (1/\tau) \sum_{k=0}^{mn-1} \sum_{i=1}^{m} \frac{(\mu_i)^{k+i}(\mu_i-1)}{\mu_i \{\mu_i\}^{m-1} - m(\mu_i-1)} \tag{34}
\]

which when substituted in (26) gives the closed-form solution for the processing time. Note that the function \( Q(s) \) may have imaginary roots. However, the imaginary part gets cancelled when the summation in (33) is performed. It can be verified that \( Q(s) \) has distinct roots. However, in cases where \( Q(s) \) has repeated roots, one can use the General Expansion Theorem for rational generating functions (given in [17]) and obtain closed-form expression for \( X_j \) as functions of the roots of \( Q(s) \). Following is an example that demonstrates the procedure explained above for the case when \( m = 2 \) and \( n = 3 \).

For this system, the generating function \( G(s) \) given by (28) can be written as,
\[
G(s) = (1 + s)/\{1 - \tau s(1 + s)\}
\]

where
\[
P(s) = 1 + s
\]
\[
Q(s) = 1 - \tau s(1 + s).
\]

The roots of the equation \( Q(s) \) are given by,
\[
(1/\mu_1) = (-1 + \sqrt{1 + 4/\tau})/2
\]
\[
(1/\mu_2) = (-1 - \sqrt{1 + 4/\tau})/2.
\]
Using (30)

\[ a_1 = \frac{(c_1 + c_2)}{((c_1 - c_2)(c_1c_2))} \]

\[ a_2 = \frac{-(c_1 - c_2)}{((c_1 + c_2)(c_1c_2))} \]

where

\[ c_1 = \sqrt{4 + \tau}, \quad c_2 = \sqrt{\tau}. \]

Using (29) we have, for different values of \( k \),

\[ x_0 = 1, \]

\[ x_1 = 1 + \tau, \]

\[ x_2 = 2\tau + \tau^2, \]

\[ x_3 = \tau + 3\tau^2 + \tau^3, \]

\[ x_4 = 3\tau^2 + 4\tau^3 + \tau^4, \]

\[ x_5 = \tau^2 + 6\tau^3 + 5\tau^4 + \tau^5 \]

\[ T(m, n) = \frac{1 + 2\tau + 4\tau^2 + 8\tau^3 + 11\tau^4 + 6\tau^5 + \tau^6}{3 + 6\tau + 12\tau^2 + 19\tau^3 + 17\tau^4 + 7\tau^5 + \tau^6}. \]

Note that in order to compute \( X_i \)s using this procedure, for high values of \( m \), one has to obtain the roots of a higher degree polynomial which is a tedious procedure. But, once the roots are obtained, computing \( X_i \)s are easy.

2) Solution Using Binomial Expansion: Consider the generating function \( G(s) \) in the form given in (28). This can be written as

\[ G(s) = (1 - s^m)/(1 - (1 + \tau)s + \tau s^{m+1}). \]

Defining

\[ y = a(1 + bs^m) \]

where

\[ a = (1 + \tau) \]

and

\[ b = -\tau/a \]

we have,

\[ G(s) = (1 - s^m)(1 - y)^{-1}. \]

Expanding (39), \( G(s) \) can be written as the difference of two series denoted by \( G_1 \) and \( G_2 \) where

\[ G_1 = 1 + y + y^2 + \ldots \]

\[ G_2 = s^m + s^m y + s^m y^2 + \ldots. \]

From (36) \( y^i \) can be written by using binomial expansion as,

\[ y^i = a^i s^i \left\{ \binom{i}{0} (bs^m)^0 + \binom{i}{1} (bs^m) + \ldots + \binom{i}{i} (bs^m)^i \right\}. \]

Thus, \( X_p \), the coefficient of \( s^p \) in (39), is obtained as the difference of the coefficient of \( s^p \) in (40) and (41), denoted by \( X_{p1} \) and \( X_{p2} \), respectively, as

\[ X_p = X_{p1} - X_{p2} \]

where

\[ X_{p1} = \sum_{k=0}^{\frac{m-1}{2}} a^{p-k} m_k b^k \left( \frac{p-km}{k} \right) \]

and

\[ X_{p2} = \sum_{k=0}^{\frac{m-1}{2}} a^{p-(k+1)m} b^k \left( \frac{p-(k+1)m}{k} \right) \]

Note that (44) and (46) can be further simplified to

\[ X_{p1} = \sum_{k=0}^{\frac{m-1}{2}} (-1)^k (1 + \tau)^{p-(m+1)k} \left( \frac{p-km}{k} \right) \]

\[ X_{p2} = \sum_{k=0}^{\frac{m-1}{2}} (-1)^k \tau^k (p-(m+1)k) \left( \frac{p-(k+1)m}{k} \right). \]

Thus, \( X_p \) can be obtained from (43) using (48) and (49). Substituting this into (24) we obtain \( X \) and subsequently using (26) we get the processing time.

Substituting \( m = 2 \) and \( n = 3 \) in the above equation we get the same solution as we do in the example given in the earlier section. This approach does not require the computation of the roots of a high degree polynomial. This is an advantage over the previous method when \( m \) is large. However, here the computation of \( X_i \)s, involves the evaluation of factorials of fairly large numbers when \( n \) is large.

B. Single-Level Tree Without Front-End

In this section we derive closed-form solutions for the case when none of the processors are equipped with front-end processors. Here (12)–(14) can be rewritten as,

\[ \alpha_{i,1} = a_{i,1}(1 + \tau)^{j-1}, \quad i = 2, \ldots, m, \]

\[ \alpha_{i,j} = \left\{ \sum_{k=1}^{j-1} a_{k,j} + \sum_{k=i+1}^{m} a_{k,j-1} \right\} \tau, \]

\[ i = 1, \ldots, m; \quad j = 2, \ldots, n, \]

\[ a_0 = a_{1,1}. \]

Following the procedure adopted in the previous section, we use (19) and (20) to obtain from (50) and (51)

\[ X_k = (1 + \tau)^k, \quad k = 0, 1, \ldots, m - 1, \]

\[ X_k = (X_{k-1} + X_{k-2} + \ldots + X_{k-m+1}) \tau, \]

\[ k = m, m + 1, \ldots, mn - 1. \]
Using the normalizing equation (10), we get
\[ a_{1,1} = 1/(1 + X) \]  
\[ (55) \]
where \( X \) is given by (24). The processing time \( T(m,n) \) is obtained as,
\[ T(m,n) = (\tau X + 1)w_{T_C}. \]  
\[ (56) \]
Here also \( X \) can be obtained using generating functions. The corresponding generating functions for the recursive equations (54) with initial conditions (53), with index of recursion suitably taken to infinity, is given by
\[ G(s) = (1 + s + \ldots + s^{m-1})/(1 - \tau(s + \ldots + s^{m-1})). \]  
\[ (57) \]
This can be expanded into a power series to obtain \( X \) as the coefficient of \( s^1 \). As mentioned earlier this can be done in two ways given below.

1) **Solution Using Rational Expansion Theorem:**
Since the degree of the numerator and denominator polynomials of (57) are same, we write \( G(s) \) as
\[ G(s) = (-1/\tau) + \{P(s)/Q(s)\} \]  
\[ (58) \]
where
\[ P(s) = 1 + (1/\tau) \]  
\[ (59) \]
\[ Q(s) = 1 - \tau(s + s^2 + \ldots + s^{m-1}). \]  
\[ (60) \]
Let the roots of the polynomial \( Q(s) \) be \( 1/\mu_1, 1/\mu_2, \ldots, 1/\mu_{m-1} \). Then we can express \( X_k \) as,
\[ X_0 = (-1/\tau) + a_1 + a_2 + \ldots + a_{m-1} \]  
\[ (61) \]
\[ X_k = a_1(\mu_1)^k + a_2(\mu_2)^k + \ldots + a_{m-1}(\mu_{m-1})^k, \]
\[ k = 1,2,\ldots, mn - 1 \]  
\[ (62) \]
where
\[ a_j = \{-\mu_jP(1/%mu_j)\}/\{Q_j'(1/%mu_j)\}, \]
\[ j = 1,\ldots, m - 1. \]  
\[ (63) \]
From which
\[ X_k = \{(1 + \tau)/\tau^2\} \sum_{i=1}^{m-1} \frac{((\mu_i)^{m+k-1}(\mu_i^-1)^2)}{((\mu_i)^{m-1} - m(\mu_i - 1))}, \]
\[ k = 1,\ldots, mn - 1. \]  
\[ (64) \]
Thus substituting in (24) we obtain,
\[ X = \{(1 + \tau)/\tau^2\} \sum_{k=0}^{mn-1} \sum_{i=1}^{m-1} \frac{((\mu_i)^{m+k-1}(\mu_i^-1)^2)}{((\mu_i)^{m-1} - m(\mu_i - 1))}, \]
\[ - (1/\tau) \]  
\[ (65) \]
which, when substituted in (56) yields the closed-form solution for the processing time.

2) **Solution Using Binomial Expansion:**
Consider the generating function \( G(s) \) in the form as given in
\[ G(s) = (1 - s^m)/(1 + \tau s^m - s(1 + \tau)). \]  
\[ (66) \]
Defining
\[ r = as(1 + bs^{m-1}) \]  
\[ (67) \]
where \( a \) and \( b \) are as defined in (37) and (38). Then,
\[ G(s) = (1 - s^m)(1 - r)^{-1}. \]  
\[ (68) \]
Expanding the RHS of this expression, \( G(s) \) can be written as the difference of two series. We denote them as \( G_1 \) and \( G_2 \) where
\[ G_1 = 1 + r + r^2 + \ldots \]  
\[ (69) \]
\[ G_2 = s^m + s^mr + s^mr^2 + \ldots \]  
\[ (70) \]
From (67), \( r^i \), i.e., the \( j \)th power of \( r \), can be written by using binomial expansion as,
\[ r^i = a(is^i) \left\{ \binom{i}{0} (bs^{m-1})^0 + \binom{i}{1} (bs^{m-1})^1 \right. \]
\[ + \left. \binom{i}{2} (bs^{m-1})^2 + \ldots + \binom{i}{j} (bs^{m-1})^j \right\}. \]  
\[ (71) \]
Since our main aim is to find the coefficient of \( s^p \), namely \( X_p \), in (68), we collect the coefficients corresponding to \( s^p \) from (69) and (70), as \( X_{p1} \) and \( X_{p2} \), respectively,
\[ X_p = X_{p1} - X_{p2} \]  
\[ (72) \]
where,
\[ X_{p1} = \sum_{k=0}^{L(p)} a_{p-k(m-1)}b^k \left( \frac{p - k(m - 1)}{k} \right) \]  
\[ (73) \]
where
\[ L(p) = \lfloor p/m \rfloor \]  
\[ (74) \]
\[ X_{p2} = \sum_{k=0}^{M(p)} a_{p-m(k+1) + k}b^k \left( \frac{p - m(k + 1) + k}{k} \right) \]  
\[ (75) \]
where
\[ M(p) = \lceil p/m \rceil - 1. \]  
\[ (76) \]
Note that (73) and (75) can be further simplified to
\[ X_{p1} = \sum_{k=0}^{L(p)} (-1)^k r^k(1 + r)^{p-k(m - 1)} \left( \frac{p - k(m - 1)}{k} \right) \]  
\[ (77) \]
\[ X_{p2} = \sum_{k=0}^{M(p)} (-1)^k r^k(1 + r)^{p-m(k+1)} \left( \frac{p - m(k + 1) + k}{k} \right). \]  
\[ (78) \]
Thus, \( \Delta \) can be obtained from (77), (78), and (72). Substituting \( \Delta \) into (24) we obtain \( X \) and subsequently using (56) we obtain the closed-form solution for the processing time.

V. ASYMPTOTIC PERFORMANCE ANALYSIS

In the previous sections we have obtained closed-form solutions for the processing time in a single-level tree network when there are \( m \) child processors and the processing load is distributed in \( n \) installments. It can be seen that the performance gets better as \( m \), or \( n \), or both increase. It is natural to examine the limit of performance enhancement that can be achieved by increasing \( m \) and \( n \). In other words, we want to obtain the following expressions:

\[
T(m,m) = \lim_{n \to \infty} T(m,n) \tag{79}
\]

\[
T(m,n) = \lim_{m \to \infty} T(m,n). \tag{80}
\]

In the following sections, we evaluate (79) and (80) for both with and without front-end cases.

A. Single-Level Tree With Front-End Processors

Consider (21) and (22) in Section IVA. These set of equations, in general, can be rewritten in a different form as,

\[
Y(i,0) = 1, \quad i = 0, 1, \ldots, m - 1 \tag{81}
\]

\[
Y(i,0) = 0, \quad i = m, m + 1, \ldots, mn - 1 \tag{82}
\]

\[
Y(k,j) = 0, \quad k < 0 \tag{83}
\]

\[
Y(i,j) = Y(i - 1, j - 1) + Y(i - 2, j - 1) + \cdots + Y(i - m, j - 1), \quad i = 0, 1, \ldots, mn - 1, \quad j = 1, 2, \ldots, mn - 1 \tag{84}
\]

where \( Y(i,j) \) is the coefficient of \( \tau^j \) in \( X \). Then, using (81)–(84), we can rewrite (24) as

\[
\Delta = \sum_{i=0}^{m-1} Y(i,0) + \sum_{i=0}^{m-1} Y(i,1) + \cdots + \sum_{i=0}^{m-1} Y(i,mn - 1) \tag{85}
\]

which can be written as

\[
\Delta = \sum_{j=0}^{m-1} P(j) \tau^j \tag{86}
\]

where

\[
P(j) = \sum_{i=0}^{m-1} Y(i,j) \tag{87}
\]

and is defined as the coefficient of \( \tau^j \) in \( X \). In order to obtain (79) we require

\[
\lim_{n \to \infty} \Delta = \sum_{j=0}^{\infty} \left\{ \tau^j \lim_{n \to \infty} P(j) \right\}. \tag{88}
\]

From (84),

\[
\lim_{n \to \infty} P(j) = \sum_{i=0}^{\infty} Y(i - 1, j - 1) + \cdots + \sum_{i=0}^{\infty} Y(i - m, j - 1) \tag{89}
\]

which, upon using (81)–(83), reduces to

\[
\lim_{n \to \infty} P(j) = m \lim_{n \to \infty} \{ P(j - 1) \}. \tag{90}
\]

However, from (81) we know that

\[
P(0) = m. \tag{91}
\]

Hence,

\[
\lim_{n \to \infty} \Delta = m \lim_{n \to \infty} \{ P(0) \}. \tag{92}
\]

This equation can be further simplified depending on the condition we impose on the factor \( m \tau \). The following are the two different cases which may arise.

Case 1 \( m \tau < 1 \).

For this case, it can be readily seen that (92) can be written as,

\[
\lim_{n \to \infty} \Delta = m/(1 - m \tau). \tag{93}
\]

Thus, the processing time is given by,

\[
T(m,\infty) = \{1/(1 + m)\} \tau T_{cp}. \tag{94}
\]

Case 2 \( m \tau \geq 1 \).

For this case, it is obvious that (92) will not converge to a finite value. Thus, it can be seen that the processing time is given by,

\[
T(m,\infty) = \{\tau/(1 + \tau)\} \tau T_{cp}. \tag{95}
\]

Now we evaluate (80). From (81)–(84), we observe that as \( m \to \infty \) (i.e., when the number of processors tends to infinity) \( X \) also tends to infinity. This means

\[
\lim_{m \to \infty} \Delta = \{\tau/(1 + \tau)\} \tau T_{cp}, \tag{96}
\]

which is the same as in Case 2 above.

Summarizing the results, we have, for the case where processors are equipped with front-ends,

\[
T(m,\infty) = \{1/(1 + m)\} \tau T_{cp}, \quad \text{if} \quad m \tau < 1 \tag{97}
\]

\[
T(m,\infty) = \{\tau/(1 + \tau)\} \tau T_{cp}, \quad \text{if} \quad m \tau \geq 1 \tag{98}
\]

where \( T(\infty,n) = \{\tau/(1 + \tau)\} \tau T_{cp} \).

B. Single-Level Tree Without Front-End Processors

Consider (53) and (54) in Section IVB. These equations, in general, can be rewritten as,
\[ Y(i,0) = 1, \quad i = 0,1,\ldots,m-1 \quad (98) \]
\[ Y(i,0) = 0, \quad i = m,m+1,\ldots,mn - 1 \quad (99) \]
\[ Y(k,j) = 0, \quad k < 0, \quad (100) \]
\[ Y(i,j) = Y(i-1,j-1) + Y(i-2,j-1) + \cdots + Y(i-m,j-1), \quad i = 0,1,\ldots,mn - 1, \quad j = 1,2,\ldots,mn - 1 \quad (101) \]

where \( Y(i,j) \) is the coefficient of \( \tau^j \) in \( X_i \). Using the same procedure as in the previous section,

\[ \lim_{n \to \infty} P(j) = \sum_{i=0}^{\infty} Y(i-1,j-1) \]
\[ + \cdots + \sum_{i=0}^{\infty} Y(i-m+1,j-1) \quad (102) \]

which upon using (98)-(100), reduces to,

\[ \lim_{n \to \infty} P(j) = (m - 1) \lim_{n \to \infty} [P(j - 1)]. \quad (103) \]

Here, from (98)

\[ P(0) = m. \quad (104) \]

Thus,

\[ \lim_{n \to \infty} X = \sum_{i=0}^{\infty} X_i = m \{1 + (m-1)\tau + (m-1)^2\tau^2 + \cdots\}. \quad (105) \]

Hence, depending on the factor \( (m-1)\tau \), we have the following cases.

**Case 1** \( (m-1)\tau < 1 \)

In this case, it can be seen that (105) can be written as

\[ \lim_{n \to \infty} X = m/\{1 - (m-1)\tau\}. \quad (106) \]

Thus, the processing time is given by,

\[ T(m,\infty) = [(1 + \tau)/\{(m + 1) - \tau(m - 1)\}] wT_{ep} \quad (107) \]

**Case 2** \( (m-1)\tau \geq 1 \)

Here, (105) will not converge to a finite value and hence we have

\[ T(m,\infty) = \tau wT_{ep}. \quad (108) \]

Now, we evaluate (80). This turns out to be

\[ \lim_{m \to \infty} T(m,n) = \tau wT_{ep}. \quad (109) \]

Thus, summarizing the results, we have

\[ T(m,\infty) = \frac{(1 + \tau)wT_{ep}}{(m + 1) - \tau(m - 1)}, \quad \text{if} \quad (m-1)\tau < 1 \]
\[ T(m,\infty) = \tau wT_{ep}, \quad \text{if} \quad (m-1)\tau \geq 1 \quad (110) \]
\[ T(\infty,n) = \tau wT_{ep}. \]

VI. DISCUSSION OF THE RESULTS

Fig. 6(a) and 6(b) shows \( T(m,n) \) versus \( n \) for a given value of \( \tau \) and different values of \( m \). The values of \( T(m,n) \) for \( n = 1 \) are the processing times obtained by using the scheme proposed in [2]. When the number of installments increase, the processing time reduces. But, the reduction is not too significant after just a few installments. In fact, the processing time saturates to the value given by \( T(m,\infty) \) in (97) and (110). This appears to be true for both the cases when the processors are equipped with and without front-ends.

It also shows that the performance is always better when the processors are equipped with front-end than when they are not for the same values of \( m \) and \( n \). This is true even if \( m \) or \( n \) goes to infinity.

In Fig. 6(c) and 6(d), \( T(m,n) \) versus \( m \) for a given \( \tau \) and different values of \( n \) are shown. As predicted by (97) and (110), as the number of processors increase, the processing time converges to the same value \( T(\infty,n) \) regardless of the value of \( n \).

The plots shown in Fig. 6 reveal an interesting tradeoff relation between the number of installments and the number of processors. When the number of existing child processors are small, adding a new processor enhances the performance more than by increasing the number of installments. Whereas, when
existing number of child processors is large, increasing the number of installments will be more beneficial than adding a new processor. Further, Fig. 6 also shows that when the number of child processors are large it is possible for a network without front-ends, using a large number of installments, to have a performance comparable to a network with front-ends using one installment.

Fig. 7 shows the ultimate performance limit $T(m, \infty) \text{ versus } m$ for different values of $r$. It shows that as long as $mr < 1$, in the case with the front-end $(m - 1)r < 1$ in the case without the front-end) the value of $T(m, \infty)$ decreases with increasing $m$ and then levels off to the value given by (95) ((107) in the case without front-end). The lower envelope of these curves in the case of processors equipped with front-end is independent of $r$ unlike the case when processors are not equipped with front-end. This is also evident from Fig. 8, which shows $T(m, \infty) \text{ versus } \tau$.

The plots given in Fig. 7 reveal the following information. If the number of child processors is large, and a large number of installments are used, then the performance achieved remains almost unchanged even when some processors are removed from the network. It implies that, in a single-level tree network, for a given value of $\tau$, the processors beyond a certain number are redundant if a sufficiently large number of installments are used.

From Fig. 8, we observe that in the case when processors are equipped with front-ends it is possible to achieve almost full utilization of the processors when $mr < 1$ by using sufficiently large number of installments. In fact, the processing time $T(m,n)$ is independent of the value of $\tau$ so long as this condition is met. This is however not true when the processors are not equipped with front-ends.

All the above results produce some interesting insights to the load distribution problem for processing time minimization. For a given network of one root processor and $m$ child processors equipped with front-ends, and with a given value of the processing time performance can be improved further by either adding more processors (which is not a cost-effective solution) or by sending the load in installments as proposed here. When $mr < 1$, by increasing the
number of installments one can ensure almost full utilization of all the processors. When $mr \geq 1$, by increasing the number of installments, one can reach a limit which is the same as the limit reached by using a large number of processors. Similar arguments hold true for the case when the processors are not equipped with front-ends.

VII. CONCLUSIONS

This paper gives a new strategy for load distribution in a single-level tree network to minimize the processing time. Some necessary conditions were stated for optimal load distribution. A closed-form solution for a special case of identical processors and identical links was derived using two techniques, one of which is useful when the number of processors is large and the other when the number of installments is large. An asymptotic performance analysis was carried out, exploring the ultimate limits of performance of the system with respect to processing time. Practical issues regarding the tradeoff between the number of processors and installments was discussed. The above analysis is valid when there is no overhead associated with each installment. However if this is not the case, then it is expected that the total overhead increases with the number of installments. In earlier studies, this overhead was neglected because the total number of installments was limited by the number of processors. However, from the results obtained here, we observe that the time performance does not improve significantly beyond the first few installments (2 or 3). It may be noted that when $mr \geq 1$ (or, $(m-1)r \geq 1$), there are processors which still remain unutilized. So, it would be beneficial to explore methods by which the waiting time of these processors can be utilized. Furthermore, optimal sequencing and optimal arrangement of processors and links, as considered in [12], have not been discussed because of the nonavailability of closed-form expression in the general case of unidentical processors and links.

REFERENCES


V. Bharadwaj received his B.Sc. degree (first rank) in physics from Madura College, Madurai, India, in 1987, the M.E. degree in electrical communication engineering from the Indian Institute of Science, Bangalore, India, in 1991, and the Ph.D. degree in engineering, from the Indian Institute of Science in 1994. Currently he is with the Department of Computer Science, Concordia University, Montreal, Canada.

His current research interests include multi-media systems, distributed computing systems, optical switching networks, and integrated optical sensors.

D. Ghose received the B.Sc. (Engg) degree in electrical engineering from the Regional Engineering College, Rourkela, India, 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, where he is presently an Assistant Professor in the Department of Aerospace Engineering. His research interests include distributed computing, guidance and control of aerospace vehicles, and game theory.

V. Mani received the B.E. degree in civil engineering from Madurai University, Madurai, India, 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, Indian Institute of Science. Since 1990 he has been an Assistant Professor at the Department of Aerospace Engineering, Indian Institute of Science. His research interests include queueing networks, reliability, neural computing, and mathematical modeling.