Distributed Computation in Linear Networks: Closed-Form Solutions

V. MANI

D. GHOSE

Indian Institute of Science

A linear network of communicating processors is analyzed. The processors in the network may or may not be equipped with front-end processors. The processing load originates either at the boundary or at the interior of the network. Closed-form solutions and computational techniques are presented for the above situations, to obtain time optimal distribution of processing loads on the processors. Some important results are proved analytically using the closed-form expressions.

Manuscript received March 7, 1992; revised February 28, 1993.

IEEE Log No. T-AES/30/2/15498.

Author's address: Department of Aerospace Engineering, Indian Institute of Science, Bangalore 560012, India.

I. INTRODUCTION

One of the main problems in distributed computation is the distribution of processing load among processors to achieve minimum processing time. This distribution of processing load depends on the network architecture, speed of the processing units, speed of communication channels, the number of processing units, and the origination point of the load. This problem is faced mainly in the analysis and design of distributed intelligent sensor networks, which is a collection of units with sensing, computing, and communicating abilities [1]. In a sense, this is a resource allocation problem in which the tradeoff between computation and communication has to be dealt with. The importance of tradeoff between computation and communication has been recognized earlier by several authors [2–4].

In [1, 5], the distribution of processing load in a linear and tree network of communicating processors is studied. In [1], recursive equations for obtaining the optimal processing loads in a linear network are developed. The cases of equal and optimal division of the processing load among the processors are discussed for networks with and without front-end processors. The cases in which the processing load originates at the boundary and at the interior of the network, and also a method of incorporating the inclusion of solution time (the time taken for the processors to report the solution back to the problem originator) is also presented. The main emphasis in the study [1, 5] is the development of recursive equations for the determination of optimal processing loads on the processors and obtaining computational solutions. In a recent study [6, 7] the recursive equations for bus-oriented architecture has been developed and a similar computational technique as given in this work has been proposed. However, no closed-form solutions were derived.

In this work, a linear network of communicating processors is considered. Our main objective is to find closed-form solutions and easily implementable computational techniques for the determination of optimal processing loads allocated to each processor. The processors may or may not be equipped with front-end processors, and the processing load may originate either at the boundary or at the interior of the network. The closed-form solutions developed here are for the case in which all the processing units have the same computation speed and all the channels have the same communication speed. If the speeds are different for different processors and channels, then the computational algorithm is useful. The advantage of the closed-form solution is that analytic techniques can be brought to bear upon them to get results which otherwise would require considerable computational effort. We use the closed-form expressions to prove the important result that in a given linear network of

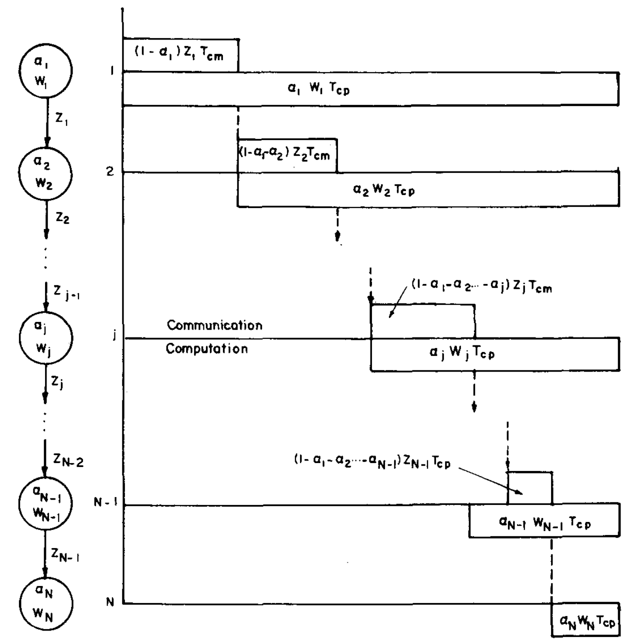


Fig. 1. (a) Timing diagram. Origination at boundary. With front-end processors.

communicating processors, when the processing load originates at the interior of the network, the minimum processing time is independent of the sequence in which the starting (root) processor shares the load with its left and right neighbors.

In Section II, we consider the case in which all the processing units are equipped with front-end processors (i.e., communication and computation can be done at the same time) while in Section III we consider the case without front-end processors (i.e., communication and computation cannot be done at the same time). In both Sections II and III, two cases of the processing load originating at the boundary and at the interior of the network are analyzed.

II. LINEAR NETWORK WITH FRONT-END PROCESSORS

A. Origination at Network Boundary

In this case, the processor at the left end of the chain receives the full processing load which is to be shared in an optimal manner by all the N processors. The first processor keeps a fraction α_1 of the processing load for itself and transmits the remaining $(1-\alpha_1)$ fraction to its right immediate neighbor (the second processor). Similarly, the jth processor keeps a fraction α_j (of the total processing load) and transmits the remaining to the $(j+1)$th processor. The last processor, upon receiving the fraction of the load, does only computation. The process and the timing diagram are shown in Fig. 1(a). In order to obtain maximum parallelism and minimum time solution all the processors must stop at the same time [1]. The starting processor should compute its fraction of the processing load during the entire processing period so that the total processing time T_t equals the processing time of the starting processor [1]. From Fig. 1(a) it can also be seen that the processing time $\alpha_j w_j T_{cp}$ of the jth processor equals the transmission time, $(1-\alpha_1-\alpha_2\ldots-\alpha_j)z_jT_{cm}$ from the jth processor to the $(j+1)$th processor plus the processing time $(\alpha_{j+1}w_{j+1}T_{cp})$ of the $(j+1)$th processor. Here T_{cm} is the time to transmit the entire measurement data over a standard channel, T_{cp} is the time it takes for one standard processor to process the entire processing load, w_js are inversely proportional to the speed of the jth processor, and z_js are inversely proportional to the channel speed between jth and $(j+1)$th processor. From Fig. 1(a) the equations (for the processing load) governing the maximum parallelism and minimum time solution are written as follows [1].

$$\alpha_j w_j T_{cp} = (1-\alpha_1-\alpha_2\ldots-\alpha_j)z_jT_{cm} + \alpha_{j+1}w_{j+1}T_{cp}, \qquad j = 1,2,\ldots,N-1. \tag{1}$$

We also have the normalizing condition

$$\alpha_1+\alpha_2+\cdots+\alpha_N = 1. \tag{2}$$

From (1) and (2), we can see that there are N linear equations in N unknowns. As they are linear equations any standard solution technique, such as Gaussian elimination, can be used to obtain the values of α_j $(j = 1,2,\ldots,N)$.

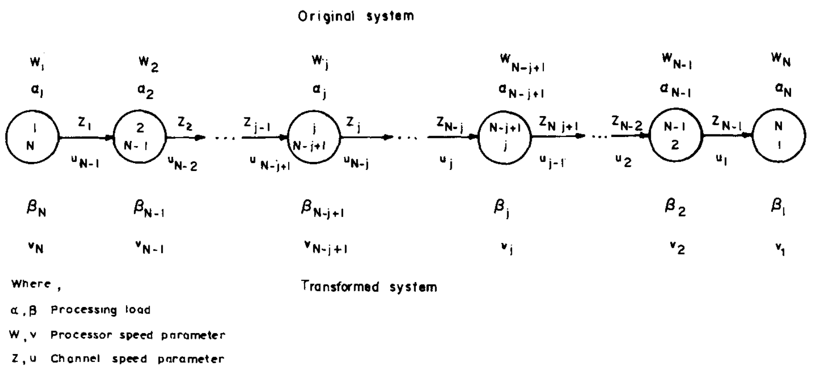


Fig. 1. (b) Parameter transformations. Origination at boundary.

TABLE I
Matrix A

j	f_j	$\sum f_j$
1	1	1
2	$\delta_1\rho+\eta_1$	$1+\delta_1\rho+\eta_1$
3	$A(2,3)\delta_2\rho+A(2,2)\eta_2$	$A(3,2)+A(2,3)$
…	…	…
j	$A(j-1,3)\delta_{j-1}\rho+A(j-1,2)\eta_{j-1}$	$A(j,2)+A(j-1,3)$
…	…	…
N	$A(N,2)$	$A(N,3)$

However, our aim is to derive a closed-form expression for α_j and also an easily implementable computational algorithm for their determination. For this purpose, we start the solution of the equation from the last processor. This approach has been used earlier, in the case of a bus-oriented architecture [6, 7]. For this, and for notational convenience, we use the following relabelling transformation.

$$\begin{aligned} \beta_j &= \alpha_{N-j+1} \qquad j=1,2,\ldots,N \\ v_j &= w_{N-j+1} \qquad j=1,2,\ldots,N \qquad (3) \\ u_j &= z_{N-j} \qquad j=1,2,\ldots,N-1. \end{aligned}$$

The transformed equations corresponding to (1) and (2) are as follows:

$$\begin{aligned} \beta_j v_j T_{cp} &= (1-\beta_1-\beta_2\ldots-\beta_{j-1})u_{j-1}T_{cm} \\ &\quad +\beta_{j-1}v_{j-1}T_{cp}, \qquad j=2,\ldots,N \qquad (4) \\ \beta_1+\beta_2+\cdots+\beta_N &= 1 \qquad (5) \end{aligned}$$

which can be rewritten as

$$\beta_j=(1-\beta_1-\beta_2\ldots-\beta_{j-1})(u_{j-1}/v_j)\rho+\beta_{j-1}(v_{j-1}/v_j), \qquad j=2,\ldots,N \qquad (6)$$

where ρ is defined as T_{cm}/T_{cp}, the ratio of communication time over the computation time. We now define $\delta_j=u_j/v_{j+1}$ and $\eta_j=v_j/v_{j+1}$, $j=1,\ldots,N-1$, and rewrite (6) as,

$$\beta_j=(\beta_1+\beta_2+\cdots+\beta_{j-1})\delta_{j-1}\rho+\beta_{j-1}\eta_{j-1}, \qquad j=2,\ldots,N. \qquad (7)$$

The parameters in the original system of equations (as developed in [1]) and the transformed system of equations used here are illustrated in Fig. 1(b). The β_js $(j=2,3,\ldots,N)$ can be expressed as functions of β_1. From the normalizing equation (5), we can compute the exact value of β_1. Once the value of β_1 is known, all the remaining $\beta_j (j=2,3,\ldots,N)$ can be obtained. For this purpose we now describe a computational algorithm.

Computational Algorithm: The set of equations shown in (7) can be solved easily in a tabular manner. For this purpose, we define a matrix A of dimension $N \times 3$, as shown in Table I. In matrix A the first column of the jth row gives the processor number, the second column of the jth row stores a number f_j which is proportional to the processing load on the jth processor, and the third column of the jth row gives the sum of all the f_js up to the jth row. The generation of the complete matrix A is shown in Table I. From Table I, it can be seen that the processing load, β_j, of the jth processor is $A(j,2)/A(N,3)$. Although it appears from Table I that it is necessary to store all the three columns, in fact it is enough to store just N values (i.e., only the second column). This algorithm can be very easily implemented as a subroutine. Note that each evaluation of $A(j,2)$ needs only three multiplications and one addition.

Actually, if only the second column is stored then the computational algorithm can be even more simplified by using the relationship

$$\begin{aligned} f_1 &= 1 \\ f_2 &= (\delta_1\rho+\eta_1)f_1 \\ f_j &= \left\{\frac{\delta_{j-1}}{\delta_{j-2}}+\delta_{j-1}\rho+\eta_{j-1}\right\}f_{j-1} \qquad (8) \\ &\quad -\frac{\delta_{j-1}\eta_{j-2}}{\delta_{j-2}}f_{j-2}, \qquad j=3,4,\ldots,N. \end{aligned}$$

This can be obtained by writing the equations for β_j and β_{j-1} from (7) and then performing some simple algebraic manipulations. This needs only the previous two entries in the second column. Now we explain how this tabular method leads to the solution of (7). For a general f_j, summation $(f_1 + f_2 + \cdots + f_{j-1})$ is available at $A(j-1,3)$, and f_{j-1} is available at $A(j-1,2)$. This process is done up to $j = N$. Note that $A(N,3)$ is the addition of $(f_1 + f_2 + \cdots + f_N)$. Thus the value of β_1 is $1/A(N,3)$. The processing load on the jth processor is thus $\beta_j = A(j,2)/A(N,3)$ for $j = 1,2,\ldots,N$. This tabular algorithm was tested with the values of w, ρ, and z given in [1] and the results obtained were found to be identical.

Closed-Form Expressions: In the previous section, an easily implementable computational technique for obtaining the processing load on individual processors was presented. In this section, we derive closed-form expressions for the processing load β_j of the jth processor under the following assumptions.

Assumption 1: All the w_js are of the same value (i.e., $\eta_j = 1$ for $j = 1,2,\ldots,N$).
Assumption 2: All the z_js are of the same value (i.e., $\delta_j = \delta$ for $j = 1,2,\ldots,N-1$).

In other words, all the processing units have the same processing speed and all the channels have the same communication speed. This is an assumption which is valid for many intelligent sensor networks. Under these assumptions, (7) representing the processing loads is rewritten as

$$\beta_j = (\beta_1 + \beta_2 + \cdots + \beta_{j-1})\nu + \beta_{j-1}, \qquad j = 2,\ldots,N \tag{9}$$

where $\nu = \delta\rho$. Now we can get a closed-form solution for (9) and the normalizing equation (5). The set of (9) are $(N-1)$ linear equations with N unknowns. So it is possible to write each β_j $(j = 1,2,\ldots,N)$ as a function of β_1 and ν. It is easy to see that, denoting $\beta_1 = \varepsilon$, this relation can be expressed as

$$\beta_j = f_j(\nu)\varepsilon \tag{10}$$

where

$$f_j(\nu) = c(j,0)\nu^0 + c(j,1)\nu^1 + \cdots + c(j,j-1)\nu^{j-1}. \tag{11}$$

The function $f_j(\nu)$ is a polynomial in ν of $(j-1)$ degree and $c(j,i)$ represents the coefficient of ν^i in the function $f_j(\nu)$ and is expressed as

$$c(j,i) = \frac{(j+i-1)!}{(2i)!(j-i-1)!}. \tag{12}$$

We can derive the above expression (12) by rewriting the recursive equation (9) as

$$\begin{aligned} &\beta_1 = \varepsilon \\ &\beta = (1+\nu)\beta \\ &\beta_j = (2+\nu)\beta_{j-1} - \beta_{j-2} \qquad j = 3,4,\ldots,N. \end{aligned} \tag{13}$$

From this it is easy to see that the coefficients in the polynomial $f_j(\nu)$ can also be expressed in a recursive manner as

$$c(j,i) = 2c(j-1,i) + c(j-1,i-1) - c(j-2,i) \tag{14}$$

with the boundary conditions

$$\begin{aligned} &c(j,0) = 1, \qquad \text{for all } j \\ &c(1,1) = 1, \\ &c(2,i) = 1, \qquad \text{for } \ i = 1,2. \end{aligned} \tag{15}$$

From (14) with the boundary conditions (15) the expressions for $c(j,i)$ given in (12) can be obtained. So the general expression for β_j can be written as

$$\beta_j = \varepsilon\left[\sum_{i=0}^{N} c(j,i)\nu^i\right]. \tag{16}$$

From the normalizing equation (5) the value of $\varepsilon = \beta_1$ can be obtained as

$$\varepsilon = \frac{1}{\sum_{j=1}^{N} f_j(\nu)} \tag{17}$$

which can be written (after some algebraic manipulations) as

$$\varepsilon = \frac{1}{\sum_{k=0}^{N-1}\left[\nu^k\left\{\sum_{j=k+1}^{N}\frac{(j+k-1)!}{(2k)!(j-k-1)!}\right\}\right]}. \tag{18}$$

An easy way of computing the value of ε without using the summation in the denominator of (18) is discussed later. An easy way to represent the closed-form solution is to write the coefficients $c(j,i)$ in the form of a matrix c shown in Table II. The columns, indexed by i, represent the powers of ν and the rows, indexed by j, represent the processor number. The non-zero entries in the matrix are given by (12). Note that for the jth processor the coefficients $c(j,i)$ are non-zero only up to $i = j-1$. It can also be seen that the entries in this table can be generated independent of the number of processors N. The fraction of the total load shared by the jth processor (when there are N processors) is given by the ratio of the polynomial in ν formed by the jth row and the sum of the polynomials formed by each row up to the Nth row, i.e.,

$$\beta_j = \frac{f_j(\nu)}{\sum_{j=1}^{N} f_j(\nu)}. \tag{19}$$

The denominator of (19) can be found easily by using the relation,

$$\sum_{j=1}^{N} f_j(\nu) = [f_{N+1}(\nu) - f_N(\nu)]/\nu \tag{20}$$

or if $f_{N+1}(\nu)$ is not available then,

$$\sum_{j=1}^{N} f_j(\nu) = f_N(\nu) + [f_N(\nu) - f_{N-1}(\nu)]/\nu. \tag{21}$$

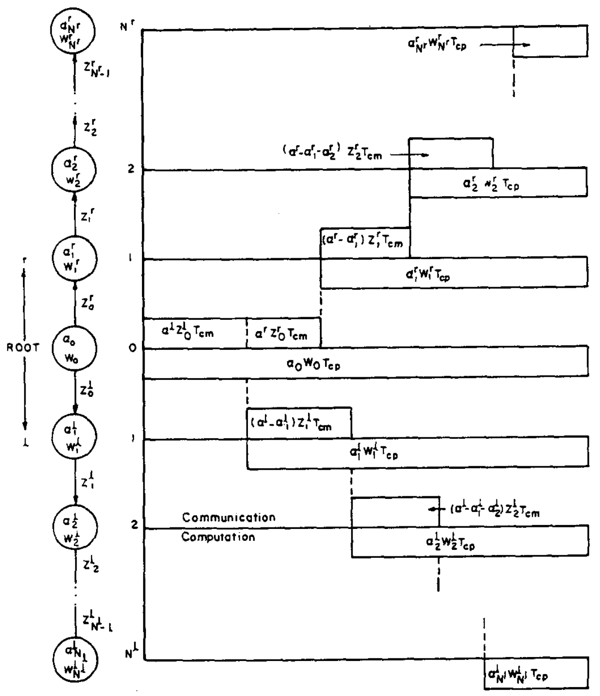


Fig. 2. (a) Timing diagram. Origination at interior. With front-end processors.

TABLE II
Matrix C

j/i	0	1	2	3	4	5	6	7	$\cdot$ $\cdot$
1	1								
2	1	1							
3	1	3	1						
4	1	6	5	1					
5	1	10	15	7	1				
6	1	15	35	28	9	1			
7	1	21	70	84	45	11	1		
8	1	28	126	210	165	66	13	1	
$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$ $\cdot$

From (21) one can easily compute the value of ε without going through the summations of all the polynomials used in (17)–(18). For example, the load on the fourth processor ($j = 4$), when there are a total of five processors ($N = 5$) is given (from Table II) by

$$\beta_4 = \frac{1 + 6\nu + 5\nu^2 + \nu^3}{5 + 20\nu + 21\nu^2 + 8\nu^9 + \nu^4}$$

and the total processing time is given by $\beta_N \nu_N T_{cp}$.

B. Origination From Network Interior

In this section, we analyze the case where a processor in the interior of the linear network, instead of the one at the end receives the full processing load. There are N^l processors to its left and N^r processors to its right. The processor which receives the full load (root processor) has to share the processing load with all the processors. The root processor first divides the processing load into smaller parts, then it transmits the fraction α^l of the total processing load to its left immediate neighbor and the fraction α^r of the total processing load to its right immediate neighbor and keeps the remaining fraction $(1 - \alpha^l - \alpha^r)$ for itself to compute. Upon receiving the data, the first processor on the left shares the total processing load α^l with its neighbors in a manner described in the earlier section. The first processor on the right performs a similar operation with the fraction α^r of the processing load. This situation and the timing diagram are shown in Fig. 2(a). Note that first the fraction of the load α^l is sent to the left and then the fraction of the load α^r is sent to the right. As mentioned earlier, in order to achieve maximum parallelism and minimum time solution all the processors must stop computing at the same time and the root processor should compute its fraction of the processing load during the entire processing period. The total processing time T_t is equal to the processing time of the root processor. The governing equations are written separately for the left and right side as follows.

Left side:

$$\alpha_0 w_0 T_{cp} = \alpha^l z_0^l T_{cm} + \alpha_1^l w_1^l T_{cp} \tag{22a}$$

$$\alpha_j^l w_j^l T_{cp} = (\alpha^l - \alpha_1^l - \alpha_2^l \ldots - \alpha_j^l) z_j^l T_{cm} + \alpha_{j+1}^l w_{j+1}^l T_{cp}, \qquad j = 1,2,\ldots,N^l - 1. \tag{22b}$$

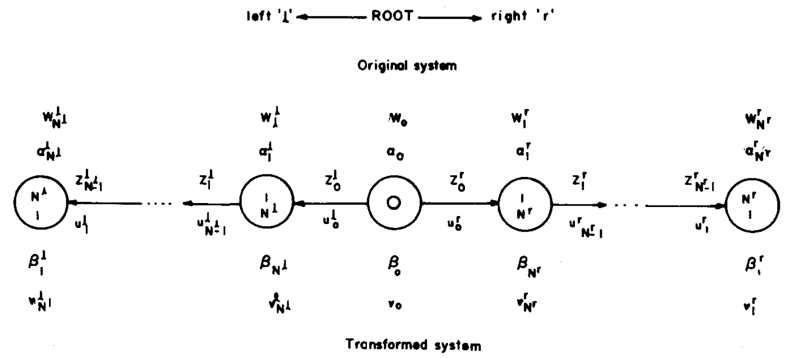


Fig. 2. (b) Parameter transformations. Origination at interior.

Right side:

$$\alpha_0 w_0 T_{cp} = (\alpha^l + \alpha^r) z_0^r T_{cm} + \alpha_1^r w_1^r T_{cp} \tag{22c}$$

$$\alpha_j^r w_j^r T_{cp} = (\alpha^r - \alpha_1^r - \alpha_2^r \ldots - \alpha_j^r) z_j^r T_{cm} + \alpha_{j+1}^r w_{j+1}^r T_{cp}, \qquad j = 1,2,\ldots,N^r - 1. \tag{22d}$$

In the above equations, the superscript identifies the direction left or right and the subscript stands for the processor number. The number of the root processor is zero. α_0 is the processing load on the root processor and is the total processing load minus the sum of the processing loads transmitted to the processors on the left and right sides. Also note that (22b) and (22d) are similar to the governing equations of the earlier section, but the sum of α_j^l $(j = 1,2,\ldots,N^l)$ is equal to α^l and the sum of α_j^r $(j = 1,2,\ldots,N^r)$ is equal to α^r. Equations (22a) and (22c) connect the processors on the left and right side. The normalizing equation is

$$\alpha_0 + \alpha_1^l + \alpha_2^l + \cdots + \alpha_N^l l + \alpha_1^r + \alpha_2^r + \cdots + \alpha_N^r r = 1. \tag{23}$$

We follow the same relabelling transformations given by (3) for the left and the right side processors with appropriate superscripts l and r. The parameters relating the root processor with its immediate neighbors, are illustrated in Fig. 2(b).

$$\begin{aligned} \eta_0^l &= v_N^l l / v_0 \\ \eta_0^r &= v_N^r r / v_0 \\ \delta_0^l &= u_0^l / v_0 \\ \delta_0^r &= u_0^r / v_0. \end{aligned} \tag{24}$$

The transformed equations are as follows.
Left side:

$$\beta_0 = (\beta_1^l + \beta_2^l + \cdots + \beta_N^l l)\delta_0^l \rho + \beta_{N^l}^l \eta_0^l \tag{25a}$$

$$\beta_j^l = (\beta_1^l + \beta_2^l + \cdots + \beta_{j-1}^l)\delta_{j-1}^l \rho + \beta_{j-1}^l \eta_{j-1}^l, \qquad j = 2,3,\ldots,N^l. \tag{25b}$$

Right side:

$$\beta_0 = (\beta_1^l + \beta_2^l + \cdots + \beta_N^l l + \beta_1^r + \beta_2^r + \cdots + \beta_N^r r)\delta_0^r \rho + \beta_{N^r}^r \eta_0^r \tag{25c}$$

$$\beta_j^r = (\beta_1^r + \beta_2^r + \cdots + \beta_{j-1}^r)\delta_{j-1}^r \rho + \beta_{j-1}^r \eta_{j-1}^r, \qquad j = 2,3,\ldots,N^r. \tag{25d}$$

The corresponding normalization equation is

$$\beta_0 + \beta_1^l + \beta_2^l + \cdots + \beta_{N^l}^l + \beta_1^r + \beta_2^r + \cdots + \beta_{N^r}^r = 1. \tag{26}$$

We can observe that except in the equation for β_0, the equation for $\beta_1^l, \beta_2^l, \ldots, \beta_{N^l}^l$ and $\beta_1^r, \beta_2^r, \ldots, \beta_{N^r}^r$ are similar to the system of equations described in the earlier section. We denote

$$\beta_1^r = \varepsilon^r, \qquad \beta_1^l = \varepsilon^l \tag{27}$$

where ε^r and ε^l are the fraction of the processing load shared by the last processor on the right-hand side (RHS) and left-hand side (LHS), respectively. Following the steps in the earlier section all the β_j^l $(j = 1,2,\ldots,N^l)$ and β_0 can be expressed as functions of $\varepsilon^l = \beta_1^l$. Likewise all the β_j^r $(j = 1,2,\ldots,N^r)$ can be expressed in terms of $\varepsilon^r = \beta_1^r$. Now we have to find a relationship between ε^l and ε^r. This relationship is obtained by equating (25a) and (25c). Once this is done, the value of ε^r can be found from the normalizing equation and the processing load on all the processors can be computed.

Computational Algorithm: The set of equations shown in (25) along with (26) can be solved using the tabular method. For this purpose, we generate two matrices A^l and A^r for the left and right side, respectively, as described in the earlier section. The matrices A^l and A^r have three columns and $(N^l + 1)$ and $(N^r + 1)$ rows, respectively. The $(N^l + 1)$th row corresponds to the root processor. The $(N^r + 1)$th row is used only to find the relation between ε^l and

ε^r. That relation is given by

$$\varepsilon^l = \frac{A^r(N^r+1,2)}{\left\{A^l(N^l+1,2)\left(1-\frac{\delta_0^r}{\delta_0^l}\right)\right\} + \left\{A^l(N^l,2)\eta_0^l\frac{\delta_0^r}{\delta_0^l}\right\}}\varepsilon^r. \tag{28}$$

This relationship between ε^l and ε^r given in (28) is obtained by equating (25a) and (25c), yielding

$$\left\{\beta_{N^l+1}^l\left(1-\frac{\delta_0^r}{\delta_0^l}\right) + \beta_{N^l}^l\eta_0^l\frac{\delta_0^r}{\delta_0^l}\right\}\varepsilon^l = \beta_{N^r+1}^r\varepsilon^r. \tag{29}$$

Now we multiply all the elements of matrix A^l by the relation given in (28) and call this new matrix $\hat{A}^l$. The value of ε^r is now obtained as

$$\varepsilon^r = \frac{1}{\hat{A}^l(N^l+1,3)+A^r(N^r,3)}. \tag{30}$$

Once the value of ε^r is known, the values of β_js can be found from

$$\begin{aligned}\beta_0 &= \frac{\hat{A}^l(N^l+1,2)}{\hat{A}^l(N^l+1,3)+A^r(N^r,3)}\\ \beta_j^l &= \frac{\hat{A}^l(j,2)}{\hat{A}^l(N^l+1,3)+A^r(N^r,3)}, \quad j=1,2,\ldots,N^l \\ \beta_j^r &= \frac{A^r(j,2)}{\hat{A}^l(N^l+1,3)+A^r(N^r,3)}, \quad j=1,2,\ldots,N^r.\end{aligned} \tag{31}$$

Closed-Form Expressions: The closed-form expressions obtained in this section are based on the assumptions that all the processing units have the same processing speed and all the channels have the same communication speed. Now (25) is written as follows.
Left side:

$$\beta_0 = (\beta_1^l+\beta_2^l+\cdots+\beta_{N^l}^l)\nu + \beta_{N^l}^l \tag{32a}$$

$$\beta_j^l = (\beta_1^l+\beta_2^l+\cdots+\beta_{j-1}^l)\nu + \beta_{j-1}^l, \quad j=2,3,\ldots,N^l. \tag{32b}$$

Right side:

$$\beta_0 = (\beta_1^l+\beta_2^l+\cdots+\beta_{N^l}^l + \beta_1^r+\beta_2^r+\cdots+\beta_{N^r}^r)\nu + \beta_{N^r}^r \tag{32c}$$

$$\beta_j^r = (\beta_1^r+\beta_2^r+\cdots+\beta_{j-1}^r)\nu + \beta_{j-1}^r, \quad j=2,3,\ldots,N^r. \tag{32d}$$

Exactly as in the previous section, the β_js can be expressed as

$$\begin{aligned}\beta_j^l &= f_j(\nu)\varepsilon^l, \quad j=1,2,\ldots,N^l\\ \beta_j^r &= f_j(\nu)\varepsilon^r, \quad j=1,2,\ldots,N^r\\ \beta_0 &= f_{N^l+1}(\nu)\varepsilon^l\end{aligned} \tag{33}$$

where $f_j(\nu)$ is a polynomial of degree $(j-1)$ as discussed in the earlier section. The coefficients of these polynomials are the same as given in (12) and represented in Table II. Now equating (32a) and (32c) we obtain

$$\beta_{N^l}^l = (\beta_1^r+\beta_2^r+\cdots+\beta_{N^r}^r)\nu + \beta_{N^r}^r \tag{34}$$

$$\varepsilon^l = \frac{f_{N^r+1}(\nu)}{f_{N^l}(\nu)}\varepsilon^r \tag{35}$$

Using (35), (33) are written as

$$\begin{aligned}\beta_0 &= \frac{f_{N^l+1}(\nu)f_{N^r+1}(\nu)}{f_{N^l}(\nu)}\varepsilon^r\\ \beta_j^l &= \frac{f_j(\nu)f_{N^r+1}(\nu)}{f_{N^l}(\nu)}\varepsilon^r, \quad j=1,2,\ldots,N^l\\ \beta_j^r &= f_j(\nu)\varepsilon^r, \quad j=1,2,\ldots,N^r.\end{aligned} \tag{36}$$

Substituting (36) in the normalizing equation (26), the value of $\varepsilon^r = \beta_1^r$ can be obtained as

$$\varepsilon^r = \frac{1}{\left\{\sum_{j=1}^{N^r} f_j(\nu)\right\} + \frac{f_{N^r+1}(\nu)}{f_{N^l}(\nu)}\left\{\sum_{j=1}^{N^l+1} f_j(\nu)\right\}}. \tag{37}$$

Note that $f_{N^r+1}(\nu)$ and $f_{N^l+1}(\nu)$ are the polynomials generated by the (N^l+1)th and (N^r+1)th rows of the matrix C (Table II) and they need not be related to any physical processors in the network. However, it should also be noted that $f_{N^l+1}(\nu)\varepsilon^l$ is also the load on the root processor. Further, the total processing time is given by $\beta_0 w_0 T_{cp}$.

As mentioned earlier, the root processor follows the sequence of first sending the fraction of the load α^l to the left and then sending the fraction of the load α^r to the right. Now we prove that the minimum computation time remains the same, even when the sequence is reversed (i.e., when the root processor first sends a processing load to the right and then sends a processing load to the left).

It can be seen that the following two situations are equivalent. 1) There are N^l processors on the left and N^r processors on the right of the root processor. The load is first sent to the right. 2) There are N^r processors on the left and N^l processors on the right of the root processor. The load is first sent to the left.

Thus the minimum time required for computing the load in case of 1 above can be found by obtaining the same for 2 above, which can be done easily by exchanging N^l and N^r in (37) and (36), and then substituting (37) into (36). This gives the value of β_0 corresponding to situation 2 as,

$$\beta_0 = \frac{f_{N^r+1}(\nu)f_{N^l+1}(\nu)}{f_{N^r}(\nu)\left\{\sum_{j=1}^{N^l} f_j(\nu)\right\} + f_{N^l+1}(\nu)\left\{\sum_{j=1}^{N^r+1} f_j(\nu)\right\}}. \tag{38}$$

We prove that this value of β_0 matches with the value of β_0 in the original sequence as given by (36) and (37). In (38) the numerator remains unchanged if the values of N^l and N^r are interchanged. We have to prove that the denominator also remains unchanged if the values of N^l and N^r are interchanged. In other words, we have to prove the following:

$$(f_{N^l}(\nu))\left\{\sum_{j=1}^{N^r} f_j(\nu)\right\} + \left\{f_{N^r+1}(\nu)\sum_{j=1}^{N^l+1} f_j(\nu)\right\} - \{f_{N^r}(\nu)\}\left\{\sum_{j=1}^{N^l} f_j(\nu)\right\} - \left\{f_{N^l+1}(\nu)\sum_{j=1}^{N^r+1} f_j(\nu)\right\} = 0. \tag{39}$$

We know that

$$\sum_{j=1}^{N} f_j(\nu) = \sum_{j=1}^{N-1} f_j(\nu) + f_N(\nu). \tag{40}$$

Using (40), the LHS of (39) reduces to

$$\left\{f_{N^l}(\nu)\sum_{j=1}^{N^r} f_j(\nu)\right\} + \left\{f_{N^r+1}(\nu)\sum_{j=1}^{N^l} f_j(\nu)\right\} - \left\{f_{N^r}(\nu)\sum_{j=1}^{N^l} f_j(\nu)\right\} - \left\{f_{N^l+1}(\nu)\sum_{j=1}^{N^r} f_j(\nu)\right\}. \tag{41}$$

Collecting the terms corresponding to N^l and N^r, expression (41) can be written as

$$\left\{\sum_{j=1}^{N^l} f_j(\nu)\right\}\{f_{N^r+1}(\nu) f_{N^r}(\nu)\} - \left\{\sum_{j=1}^{N^r} f_j(\nu)\right\}\{f_{N^l+1}(\nu) - f_{N^l}(\nu)\}. \tag{42}$$

In (20), putting $N = N^r$ and $N = N^l$, and substituting the resultant expressions in (42), the LHS of (39) reduces to zero. Hence the minimum computation time is independent of the sequence of load distribution by the root processor. However, the load on the individual processors are dependent on the sequence.

III. LINEAR NETWORK WITHOUT FRONT-END PROCESSORS

A. Origination at Network Boundary

In this case the processor at the left end of the chain receives the full processing load, which is to be shared in an optimal manner by all the N processors. The processors are not equipped with front-end processors, so that the processors cannot compute and communicate at the same time. The first processor keeps a fraction α_1 of the processing load for itself. First it transmits the remaining processing load $(1-\alpha_1)$ fraction to its right immediate neighbor (the second processor) and then computes its own fraction α_1. Similarly, the jth processor, upon receiving the fraction of the load $(1-\alpha_1-\alpha_2-\cdots-\alpha_{j-1})$ keeps a fraction α_j of the processing load for its computation. First it transmits $(1-\alpha_1-\alpha_2\ldots-\alpha_j)$ to the $(j+1)$th processor and then computes its fraction α_j. The process and the timing diagram are shown in Fig. 3. In this case also, in order to obtain maximum parallelism and minimum time solution, all the processors must stop at the same time [1]. From Fig. 3, it can be seen that the processing time $\alpha_j w_j T_{cp}$ of the jth processor equals the transmission time $(1-\alpha_1-\alpha_2-\cdots-\alpha_{j+1})z_{j+1}T_{cm}$ to the $(j+1)$th processor plus the processing time $\alpha_{j+1}w_{j+1}T_{cp}$ of the $(j+1)$th processor. From Fig. 3 the equations for the processing loads governing maximum parallelism and minimum time solution are written as follows.

$$\alpha_j w_j T_{cp} = (1-\alpha_1-\alpha_2-\cdots-\alpha_{j+1})z_{j+1}T_{cp} + \alpha_{j+1}w_{j+1}T_{cp}, \qquad j = 1,2,3,\ldots,N-1. \tag{43}$$

With the normalizing condition

$$\alpha_1+\alpha_2+\cdots+\alpha_N = 1. \tag{44}$$

Equations (43) and (44) are of the same type as discussed in the case with front-end processors. An interesting point to note from these equations is that, without front-end processors, the computation time of the last Nth processor is equal to the computation time of the $(N-1)$th processor. The reason is that the Nth processor need not communicate and so both the Nth and $(N-1)$th processors start computing at the same time.

Now we follow the same analysis and relabelling transformations (3) of the earlier section. The only difference is that in this case, the δ_js are defined as u_j/v_{j+2} $(j = 1,2,\ldots,N-2)$. The processing load β_j on the jth processor is

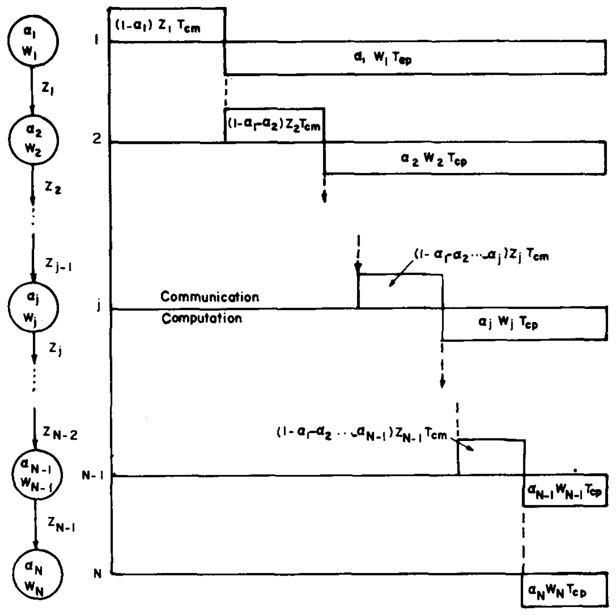
$$\beta_2 = \beta_1\eta_1$$

$$\beta_j = (\beta_1+\beta_2+\cdots+\beta_{j-2})\delta_{j-2}\rho + \beta_{j-1}\eta_{j-1}, \qquad j = 3,4,\ldots,N \tag{45}$$

$$\beta_1+\beta_2+\cdots+\beta_N = 1. \tag{46}$$

Now we describe the tabular method for computation of the processing load β_j $(j = 1,2,\ldots,N)$, in the case without front-end processors.

Computational Algorithm: Table III (matrix B) is developed in the same manner as matrix A in the case with front-end processors. The differences are 1) the



α_1 W_1
z_1
α_2 W_2
z_2
z_{j-1}
α_j W_j
z_j
z_{N-2}
α_{N-1} W_{N-1}
z_{N-1}
α_N W_N
1
2
j
N-1
N
$(1-\alpha_1)\, z_1\, T_{cm}$
$\alpha_1\, W_1\, T_{cp}$
$(1-\alpha_1-\alpha_2)\, z_2 T_{cm}$
$\alpha_2\, W_2\, T_{cp}$
$(1-\alpha_1-\alpha_2 \cdots -\alpha_j) z_j\, T_{cm}$
Communication
Computation
$\alpha_j\, W_j\, T_{cp}$
$(1-\alpha_1-\alpha_2 \cdots -\alpha_{N-1}) z_{N-1}\, T_{cm}$
$\alpha_{N-1} W_{N-1} T_{cp}$
$\alpha_N W_N T_{cp}$

Fig. 3. Timing diagram. Origination at boundary. Without front-end processors.

TABLE III
Matrix B

j	g_j	$\sum g_j$
1	1	1
2	η_1	$1+\eta_1$
3	$B(1,3)\delta_1\rho + B(2,2)\eta_2$	$B(3,2)+B(2,3)$
...	...	...
j	$B(j-2,3)*\delta_{j-2}\rho$ $+B(j-1,2)*\eta_{j-1}$	$B(j,2)+B(j-1,3)$
...	...	...
N	$B(N,2)$	$B(N,3)$

initial values are $B(1,2)=1$, $B(1,3)=1$, $B(2,2)=\eta_1$, and $B(2,3)=1+\eta_1$, and 2) the elements of $B(j,2)$ are a function of $B(j-2,3)$ and $B(j-1,2)$ whereas $A(j,2)$ is a function of $A(j-1,3)$ and $A(j-1,2)$. From Table II it can be seen that the processing load β_j of the jth processor is $B(j,2)/B(j,N)$.

Closed-Form Expressions: In this section, we derive closed-form expressions for the processing load β_j of the jth processor, when the processors are not equipped with front-end processors. We follow the same assumption that all the processing units have the same processing speed and all the channels have the same communication speed. With these assumptions, (45) is written as

$$\begin{aligned}&\beta_2 = \beta_1\\ &\beta_j = (\beta_1+\beta_2+\cdots+\beta_{j-2})\nu + \beta_{j-1}\end{aligned} \tag{47}$$

where $\nu = \delta\rho$. Now we can get a closed-form solution for (47) with the normalizing equation (46). For this purpose, we follow the same idea used in the case with front-end processors. By denoting $\beta_1 = \beta_2 = \varepsilon$, each $\beta_j (j=3,4,\ldots,N)$ can be expressed as

$$\beta_j = g_j(\nu)\varepsilon \tag{48}$$

where $g_j(\nu)$ is a polynomial in ν defined by

$$g_j(\nu) = [D(j,0)\nu^0 + D(j,1)\nu^1 + \cdots + D(j,L)\nu^L] \tag{49}$$

where L is $(j-1)$ divided by 2 and truncated. The coefficients $D(j,i)$ are

$$D(j,i) = \frac{(j-1)!}{(2i)!(j-2i-1)!}. \tag{50}$$

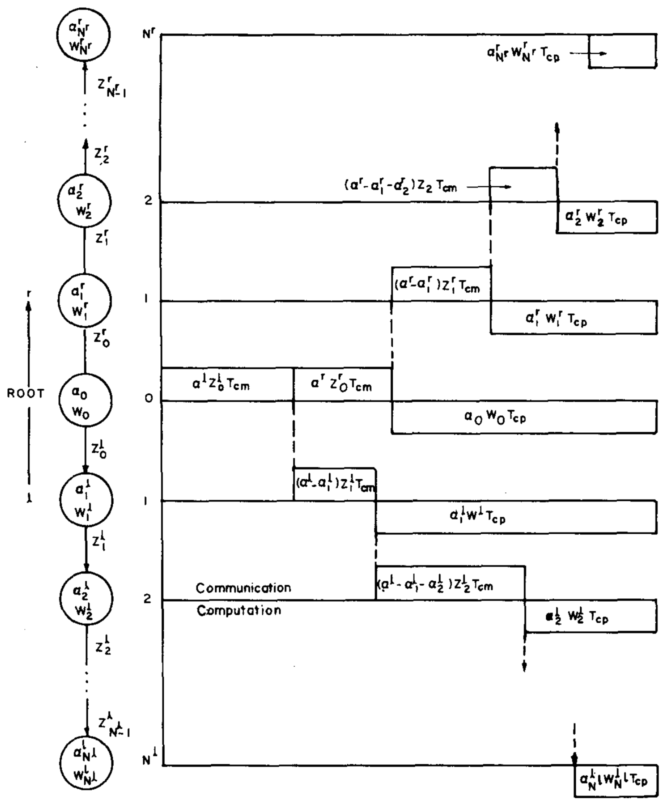
The above expression (50) is obtained in the same manner as $c(j,i)$ given in (12). The general expression for β_j is

$$\beta_j = \left[\sum_{i=1}^{L} D(j,i)\nu^i\right]\varepsilon. \tag{51}$$

From the normalizing equation (46) the value of ε can be obtained as

$$\varepsilon = \frac{1}{\sum_{k=0}^{L}\left[\nu^k\left\{\sum_{j=2k+1}^{N}\frac{(j-1)!}{(2i)!(j-2i-1)!}\right\}\right]}. \tag{52}$$

The coefficients $D(j,i)$ can be written in the form of a matrix shown in Table IV. Once this matrix is generated, the solution procedure is the same as discussed in the case with front-end processors. For



ROOT
Communication
Computation
$a^r_{N^r} W^r_{N^r} T_{cp}$
$(a^r - a^r_1 - a^r_2) z_2 T_{cm}$
$a^r_2 W^r_2 T_{cp}$
$(a^r - a^r_1) z^r_1 T_{cm}$
$a^r_1 W^r_1 T_{cp}$
$a^l z^l_0 T_{cm}$
$a^r z^r_0 T_{cm}$
$a_0 W_0 T_{cp}$
$(a^l - a^l_1) z^l_1 T_{cm}$
$a^l_1 W^l T_{cp}$
$(a^l - a^l_1 - a^l_2) z^l_2 T_{cm}$
$a^l_2 W^l_2 T_{cp}$
$a^l_{N^l} W^l_{N^l} T_{cp}$

Fig. 4. Timing diagram. Origination at interior. Without front-end processors.

TABLE IV
Matrix D

$j \backslash i$	0	1	2	3	4	5	·	·	·	·
1	1									
2	1									
3	1	1								
4	1	3								
5	1	6	1							
6	1	10	5							
7	1	15	15	1						
8	1	21	35	7						
9	1	28	70	28	1					
10	1	36	126	84	9					
·	·	·	·	·	·	·	·	·	·	·

example, the load on the fourth processor ($j = 4$), when there are total of ten processors ($N = 10$) in the network, is given by

$$\beta_4 = \frac{1 + 3\nu}{10 + 120\nu + 252\nu^2 + 120\nu^9 + 10\nu^4}.$$

The denominator of (52) can be easily generated using the following expression.

$$\sum_{j=1}^{N} g_j(\nu) = \{g_{N+2}(\nu) - g_{N+1}(\nu)\}/\nu. \qquad (53)$$

If polynomials up to $g_N(\nu)$ only are available then,

$$\sum_{j=1}^{N} g_j(N) = \left(1 + \frac{1}{\nu}\right) g_N(\nu) + \left(1 - \frac{1}{\nu}\right) g_{N-1}(\nu). \qquad (54)$$

The total processing time is given by $\beta_N \nu_N T_{cp} + (1 - \beta_N) u_{N-1} T_{cm}$.

B. Origination from Network Interior

In this section, we analyze the situation where a processor in the interior of the network receives the full processing load. There are N^l processors to its left and N^r processors to the right. The processor (root processor) which receives the full processing load, first divides the processing load into smaller parts, then it first transmits the fraction α^l of the total processing load to its left immediate neighbor and then transmits the fraction α^r to its right immediate neighbor. The root processor keeps the fraction $(1 - \alpha^l - \alpha^r)$ for its computation. Since the processors are not equipped with front-end processors, they first communicate the processing load to their neighbors and then start their own computation. The situation and the timing diagram are shown in Fig. 4. Note that the root

processor follows the sequence of first sending the load to the left and then sending the load to the right. Following the earlier section the governing equations for the left and right side are as follows.
Left side:

$$\alpha_j^l w_j^l T_{cp} = (\alpha^l - \alpha_1^l - \alpha_2^l \ldots - \alpha_{j+1}^l) z_{j+1}^l T_{cm} + \alpha_{j+1}^l w_{j+1}^l T_{cp}, \qquad j = 1,2,\ldots,N^l - 1 \tag{55a}$$

$$\alpha_0 w_0 T_{cp} = (\alpha^l - \alpha_1^l) z_1^l T_{cp} + \alpha_1^l w_1^l T_{cp} - \alpha_1^r z_1^r T_{cm} \tag{55b}$$

$$\alpha_0 w_0 T_{cp} = (\alpha^r - \alpha_1^r - \alpha_2^r \ldots - \alpha_{j+1}^r) z_{j+1}^r T_{cm} + \alpha_{j+1}^r w_{j+1}^r T_{cp}, \qquad j = 1,2,\ldots,N^r - 1 \tag{55c}$$

$$\alpha_0 w_0 T_{cp} = (\alpha^r - \alpha_1^r) z_1^r T_{cm} + \alpha_1^r w_1^r T_{cp}. \tag{55d}$$

The normalizing equation is

$$\alpha_0 + \alpha_1^l + \alpha_2^l + \cdots + \alpha_{N^l} + \alpha_1^r + \alpha_2^r + \cdots + \alpha_{N^r} = 1. \tag{56}$$

We follow the relabelling transformations given in (24) but with the following modification $\delta_j^l = u_j^l / v_{j+2}^l$, $j = 1,2,\ldots,N^l - 1$, and $\delta_j^r = u_j^r / v_{j+2}^r$, $j = 1,2,\ldots,N^r - 1$. The transformed equations are as follows.
Left side:

$$\beta_2^l = \beta_1^l \eta_1^l$$
$$\beta_j^l = (\beta_1^l + \beta_2^l + \cdots + \beta_{j-2}^l)\delta_{j-2}^l \rho + \beta_{j-1}^l \eta_{j-1}^l, \qquad j = 3,4,\ldots,N^l \tag{57a}$$

$$\beta_0 = (\beta_1^L + \beta_2^L + \cdots + \beta_{N^L-1}^L)\frac{u_{N^L-1}^l}{v_0}\rho + \beta_{N^L}^L \frac{v_{N^L}^l}{v_0} - \beta^r \frac{u_0^r}{v_0}. \tag{57b}$$

Right side:

$$\beta_2^r = \beta_1^r \eta_1^r$$
$$\beta_j^r = (\beta_1^r + \beta_2^r + \cdots + \beta_{j-2}^r)\delta_{j-2}^r \rho + \beta_{j-1}^r \eta_{j-1}^r \tag{57c}$$

$$\beta_0 = (\beta_1^r + \beta_2^r + \cdots + \beta_{N^r-1}^r)\frac{u_{N^r-1}^r}{v_0}\rho + \beta_{N^r}^r \frac{v_{N^r}^r}{v_0}. \tag{57d}$$

The normalization equation is

$$\beta_0 + \beta_1^r + \beta_2^r + \cdots + \beta_{N^r}^r + \beta_1^l + \beta_2^l + \cdots + \beta_{N^l}^l = 1. \tag{58}$$

Now we denote

$$\beta_1^l = \varepsilon^l, \qquad \beta_1^r = \varepsilon^r. \tag{59}$$

Using the above, all β_j^l ($j = 1,2,\ldots,N^l$) can be expressed as functions of ε^l and all the β_j^r ($j = 1,2,\ldots,N^r$) can be expressed as functions of ε^r. Now using the equation for β_0, we find a relationship between ε^l and ε^r. For this, we define $v_0 = v_{N^r+1}^r = v_{N^l+1}^l$ and $u_0^r = u_N^r r$, $v_0^r = v_N^r r$. Substituting these in (57b) and (57d) we get the following.
Left side:

$$\beta_0 = (\beta_1^l + \beta_2^l + \cdots + \beta_{N^l-1}^l)\frac{u_{N^l-1}^l}{v_{N^l-1}^l}\rho + \beta_{N^l}^l \frac{v_{N^l}^l}{v_{N^l+1}^l} - \beta^r \frac{u_{N^r}^r}{v_{N^r+1}^r}\rho. \tag{60a}$$

Right side:

$$\beta_0 = (\beta_1^r + \beta_2^r + \cdots + \beta_{N^r-1}^r)\frac{u_{N^r-1}^r}{v_{N^r+1}^r}\rho + \beta_{N^r}^r \frac{v_{N^r}^r}{v_{N^r+1}^r}. \tag{60b}$$

Equating (60a) and (60b) we get,

$$\beta_{N^l}^l \eta_{N^l}^l + (\beta_1^l + \beta_2^l + \cdots + \beta_{N^l}^l)\delta_{N^l-1}^l \rho - \beta_{N^l}^l \delta_{N^l-1}^l \rho = (\beta_1^r + \beta_2^r + \cdots + \beta_{N^r}^r)a + \beta_{N^r}^r \eta_{N^r}^r + (\beta_1^r + \beta_2^r + \cdots + \beta_{N^r}^r)\delta_{N^r-1}^r \rho - \beta_{N^r}^r \delta_{N^r-1}^r \rho \tag{61}$$

where $a = (u_{N^r}^r / v_{N^r+1}^r)\rho$.

In (61), all the quantities in the LHS can be expressed as a function of ε^l and all the quantities in the RHS can be expressed as a function of ε^r. Hence the relationship between ε^l and ε^r is obtained as

$$\varepsilon^l = \frac{(v_{N^r}^r - u_{N^r-1}^r \rho)\beta_{N^r}^r + \rho(u_0^r + u_{N^r-1}^r)\sum_{j=1}^{N^r}\beta_j^r}{(v_{N^l}^l - u_{N^l-1}^l \rho)\beta_{N^l}^l + \rho u_{N^l-1}^l \sum_{j=1}^{N^l}\beta_j^l}\varepsilon^r \tag{62}$$

where u_0^r is inversely proportional to the channel speed between the root processor and the first processor on the right side. Now we describe the computational algorithm for determination of the processing loads.

Computational Algorithm: The set of equations (57) along with (58) can be solved using a tabular method. For this purpose, we generate two matrices B^l and B^r for left and right side as decribed earlier (Table III, matrix B). Matrix B^l has N^l rows and matrix B^r has $N^r + 1$ rows. The $(N^r + 1)$th row corresponds to the root processor. The relation between ε^l and ε^r of (62) is obtained as

$$\frac{\varepsilon^l}{\varepsilon^r} = \left\{ \frac{(v_{N^r}^r - u_{N^r-1}^r \rho)B^r(N^r,2) + (u_0^r + u_{N^r-1}^r)B^r(N^r,3)}{(v_{N^l}^l - u_{N^l-1}^l \rho)B^l(N^l,2) + \rho u_{N^l-1}^l B^l(N^l,3)} \right\}. \tag{63}$$

Now multiplying all the elements of matrix B^l by the value $\varepsilon^l/\varepsilon^r$ obtained from (63), we obtain the matrix

$\hat{B}^l$. Then the value of ε^r can be obtained as

$$\varepsilon^r = \frac{1}{\hat{B}^l(N^l,3) + B^r(N^r+1,3)}. \tag{64}$$

Once the value of ε^r is known, the values of β_0, β_j^l $(j = 1,2,\ldots,N^l)$ and β_j^r $(j = 1,2,\ldots,N^r)$ are given by

$$\begin{aligned}
\beta_0 &= \frac{B^r(N^r+1,2)}{\hat{B}^l(N^l,3) + B^r(N^r+1,3)} \\
\beta_j^l &= \frac{\hat{B}^l(j,2)}{\hat{B}^l(N^l,3) + B^r(N^r+1,3)}, \\
&\qquad j = 1,2,\ldots,N^l \\
\beta_j^r &= \frac{B^r(j,2)}{\hat{B}^l(N^l,3) + B(N^r+1,3)}, \\
&\qquad j = 1,2,\ldots,N^r.
\end{aligned} \tag{65}$$

Closed-Form Expressions: If all the processing units have the same speed and all the channels have the same communication speed, we can derive closed-form expressions for the processing loads when the total processing load originates at the interior of the linear network. Under this condition (57) is rewritten as follows.
Left side:

$$\beta_2^l = \beta_1^l \tag{66a}$$

$$\beta_j^l = (\beta_1^l + \beta_2^l + \cdots + \beta_{j-2}^l)\nu + \beta_{j-1}^l, \qquad j = 3,4,\ldots,N^l \tag{66b}$$

$$\beta_0 = (\beta^l - \beta^r - \beta_{N^l}^l)\nu + \beta_{N^l}^l. \tag{66c}$$

Right side:

$$\beta_2^r = \beta_1^r \tag{66d}$$

$$\beta_j^r = (\beta_1^r + \beta_2^r + \cdots + \beta_{j-2}^r)\nu + \beta_{j-1}^r, \qquad j = 3,4,\ldots,N^r \tag{66e}$$

$$\beta_0 = (\beta^r - \beta_{N^r}^r)\nu + \beta_{N^r}^r. \tag{66f}$$

It is easy to represent (66) using (59) and (48) as

$$\begin{aligned}
\beta_0 &= g_{N^r+1}(\nu)\varepsilon^r \\
\beta_j^r &= g_j(\nu)\varepsilon^r, \qquad j = 1,2,\ldots,N^r \\
\beta_j^l &= g_j(\nu)\varepsilon^l, \qquad j = 1,2,\ldots,N^l.
\end{aligned} \tag{67}$$

The polynomial $g_j(\nu)$ is defined by (49) and the coefficients of the polynomial are given in Table IV (D matrix). Now equating (66c) and (66f) we obtain the relationship between ε^l and ε^r as

$$\varepsilon^l = \frac{\left\{(1-\nu)g_{N^r}(\nu) + 2\nu\sum_{j=1}^{N^r} g_j(\nu)\right\}}{\left\{(1-\nu)g_{N^l}(\nu) + \nu\sum_{j=1}^{N^l} g_j(\nu)\right\}}\varepsilon^r. \tag{68}$$

Now using (67) and (68) with the normalizing equation (58) the value of ε^r is obtained as

$$\varepsilon^r = \frac{1}{\sum_{j=1}^{N^r+1} g_j(\nu) + \left\{\dfrac{(1-\nu)g_{N^r}(\nu) + 2\nu\sum_{j=1}^{N^r} g_j(\nu)}{(1-\nu)g_{N^l}(\nu) + \nu\sum_{j=1}^{N^l} g_{N^l}(\nu)}\right\}\left\{\sum_{j=1}^{N^l} g_j(\nu)\right\}}. \tag{69}$$

Once the value of ε^r is known all the β_js can be computed from (67). The total processing time is given by $\beta_0 w_0 T_{cp} + \beta^r u_0^r T_{cm} + \beta^l u_0^l T_{cm}$.

As mentioned earlier, the root processor follows the sequence of first sending the fraction of the load α^l to the left and then sending the fraction α^r to the right. Using a similar line of reasoning to that which was employed in the situation where there were front-end processors, one can also show that the minimum computation time remains the same even when the sequence is reversed (i.e., the root processor first sends the processing load to the right and then sends the processing load to the left).

Based on the results from Sections II and III, we can state the following theorem, which is true for both the cases of with and without front-end processors.

THEOREM. *In a given linear network of communicating processors, with identical processors and communication channels, when the processing load originates at the interior of the network, the minimum computation time is independent of the sequence in which the root processor shares its load with its left and right neighbors.*

IV. CONCLUSIONS

Closed-form solutions and computational techniques are presented for the determination of optimal processing loads on the processors in a linear network. Results for all the cases, with and without front-end processors, and origination of the processing load at the boundary and at the interior of the network, are presented. It is interesting to note that in a given linear network, the minimum processing time, for the processing load originating at the interior of the network, is independent of the sequence of the load distribution by the root processor.

The advantage of the computational technique presented is that it can be easily implemented. The matrices and tables presented for computational techniques and closed-form solutions are independent of the number of processors. Hence, once the tables and matrices are generated for a large number of processors, the optimal load distribution to processors in any linear network can be found. The closed-form solution presented here is superior to previously existing methods when identical channels and identical processors are used in a linear network. If this is not the case then the computational procedure described here or in [1] should be followed. Some important conclusions which could be drawn from the computational results presented in [1] are analytically derived here from the closed-form expressions. It seems possible that the techniques used here for obtaining these closed-form expressions can be extended to obtain similar expressions for other complex networks of processors.

REFERENCES

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

[2] Tong, R. M., Tse, E., and Wishner, R. P. (1981)
Distributed hypothesis formation in sensor fusion systems.
In *Proceedings of the 20th IEEE Conference on Decision and Control*, December 1981, 1414–1424.

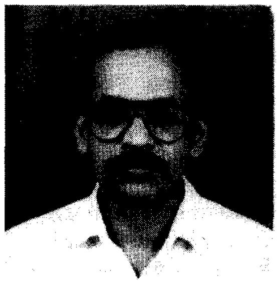
[3] Lint, B., and Agerwala, T. (1981)
Communication issues in the design and analysis of parallel algorithms.
IEEE Transactions on Software Engineering, **SE-7**, 2 (Mar. 1981), 174–188.

[4] Quinn, M. J. (1987)
Designing efficient algorithms for parallel computers.
New York: McGraw-Hill, 1987.

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

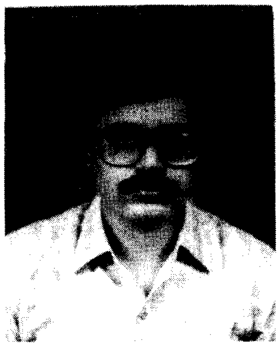
[6] Bataineh, S., and Robertazzi, T. G. (1991)
Bus oriented load sharing for a network of sensor driven processors.
IEEE Transactions on Systems, Man and Cybernetics, **SMC-21**, 5 (Sept.–Oct.), 1991, 1202–1205.

[7] Bataineh, S., and Robertazzi, T. G. (1991)
Bus oriented load sharing for a network of Intelligent sensors.
Presented at the Conference on Information Sciences and Systems, The Johns Hopkins University, Baltimore, MD, Mar. 1991.



V. Mani received the B.E. degree in civil engineering from Madurai University, 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. Since 1989 he has been with the Department of Aerospace Engineering at the Indian Institute of Science where he is currently an Assistant Professor. His research interests include queueing networks, reliability, neural computing, and mathematical modeling.



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 are in the areas of guidance and control, dynamic game theory, and distributed computing.