Performance Limits for Processor Networks with Divisible Jobs

SAMEER BATAINEH
Jordan University of Science and Technology

THOMAS G. ROBERTAZZI, Senior Member, IEEE
University at Stony Brook

Ultimate performance limits to the aggregate processing speed of networks of processors that are processing a divisible job are described. These take the form of either closed-form expressions or numerical procedures to calculate the equivalent processing speed of an infinite number of processors. These processors are interconnected in either a linear daisy chain with load origination from the network interior or a tree topology. The tree topology is particularly general as a natural way to perform load distribution in a processor network topology with cycles (e.g., hypercube, toroidal network) is to use an embedded spanning tree. Such limits on performance are important as they provide an ideal baseline against which to compare the performance of finite configurations of processors.

Manuscript received November 27, 1995; revised May 23, 1996.

IEEE Log No. T-AES/33/4/06843.

This work is supported by the BMDO/IST under the Office of the Naval Research under Grant N00014-91-J4063.

Authors' addresses: S. Bataineh, Dept. of Electrical Engineering, Jordan University of Science and Technology, Irbid, P.O. Box 3030, Jordan; T. G. Robertazzi, Dept. of Electrical Engineering, University at Stony Brook, Stony Brook, NY 11794.

I. INTRODUCTION

The problem of scheduling a number of jobs among n processors in order to minimize the finish time has received a great deal of attention [5–12, 28–32]. This previous work involved the paradigm of indivisible jobs. Under this paradigm a job can be processed by at most one processor. Jobs that can be assigned to multiple processors have received less attention [25–27]. A new paradigm of divisible jobs was discussed in [1–4, 13, 14, 16–24, 33–35]. Here the authors examined the case where a job can be partitioned into smaller fractions that can be processed independently on different processors in a multiprocessor system. Typical applications involve the processing of very large data files as in signal processing, Kalman filtering, image processing and sensor networks.

Optimal divisible job load allocation has been examined to date in a number of interconnection topologies including linear daisy chains, trees, and buses. In a linear daisy chain there is a communication link between processor i and $i + 1$ for $i = 1, 2, 3 \ldots$. Trees are, naturally, connected and acyclic graphs. The bus architecture is equivalent to a single level tree (a root node with a number of children nodes) where all links have the same transmission speed.

In [1] recursive expressions for calculating the optimal processor load allocation for linear daisy chains of processors were presented. These are based on the simplifying premise that for an optimal allocation of load, all processors must stop processing at the same time [20, 21]. Analogous solutions have been developed for tree networks [2], bus networks [3, 4], hypercubes [33], and two-dimensional meshes [34]. The equivalence of first distributing load either to the left or to the right from a point in the interior of a linear daisy chain is demonstrated in [16]. Optimal sequences of load distribution are described in [18, 19, 23, 24]. Closed-form solutions for homogeneous bus and tree networks appear in [22]. Real time systems are considered in [35]. Asymptotic results appear in [13, 14, 17]. In particular, Mani and Ghose in [17] examine performance for an infinite number of processors for linear daisy chains with load origination at a boundary processor and also single level tree networks. They do not treat the subject of this work: load origination at the interior of a linear daisy chain with an infinite number of processors or multilevel tree networks with an infinite number of processors.

For a finite number of processors, the problem of determining the amount of data that has to be assigned to each processor to achieve the minimum finish time was discussed in [1–4]. Here "finish" time is the time when all of the processors finish their computation on the fractions of the job they have been allocated. The result of this past work is to indicate that all

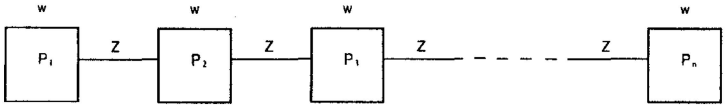


Fig. 1. Daisy chain network.

processors must stop computing at the same time to achieve the minimal finish time. Intuitively this is because if all the processors did not stop at the same time then some processors would be idle while others were busy. Certainly in this case the finish time could be improved by transferring load from busy processors to idle processors. Formal proofs of this appear in [20, 21].

The minimum finish time was found to decrease as the number of processors increases. However it was also found that after a certain number of processors were added to the network the amount of improvement diminishes. In that case, it may be advisable not to add more processors (hardware) to the chain since the cost of doing so may not be worth the small improvement in performance. In this work the problem of determining the performance of an infinite number of processors in certain cases is considered. This makes it possible to obtain the ultimate performance limit for a specific interconnection topology and load distribution sequence. Given the ultimate processing time, one can design a system by finding the number of processors that are needed in order to achieve a certain percentage of the ultimate processing time.

A simple and well-known method used in infinite electric circuit theory [15] is used here to determine the speed of an "equivalent" processor that can be used to replace an entire infinite network. Suppose, for example, that the processor at the left end of the chain, processor 1 in Fig. 1, receives a burst of measurement data and is to share it with an infinite number of network processors. The basic idea [14] is to collapse processors $2,3\ldots\infty$ into a single equivalent processor of unknown speed. However one can write an implicit equation to describe the equivalent processor that would replace the first processor and the equivalent processor for processors $2,3\ldots\infty$. This equation can be solved as the two equivalent processors it involves (one for processors $1,2\ldots\infty$ and one for processors $2,3\ldots\infty$) should have identical speeds as they both involve an infinite number of processors.

This work presents new results on the processing of a divisible job on a multiprocessor system. Specifically the case of load origination at the interior of a linear daisy chain with an infinite number of processors and the case of load origination at the root of symmetric and homogeneous tree networks with an infinite number of processors is discussed. The study of load distribution in a tree type topology is quite general in that a natural way to distribute load in topologies with cycles (e.g., hypercube, toroidal network) is through the use of an embedded spanning tree.

This work is organized as follows. Section II discusses the linear daisy chain network where load originates at a processor in the interior of the network. Systems with and without front-end processors are analyzed. Section III discusses tree networks with and without front-end processors. Section IV presents performance evaluation results. The conclusion appears in Section V.

II. LINEAR DAISY CHAIN

A. Introduction

Consider a linear daisy chain of processors as in Fig. 1. It is a network where there is a communication link from processor i to processor $i+1$ for $i = 1,2,3\ldots$. Each processor has the same computational speed $1/w$, and the channel speed between any two adjacent processors is $1/Z$. A burst of data is received by one of the processors in the chain. This data can be partitioned and the fragments distributed among the processors in order to achieve a minimum "finish time" through parallel processing. The finish time is the time when all processors are finished processing the load. The ultimate (minimum) finish time limit T^{∞} is achieved if there are an infinite number of processors in the chain.

The following definitions for some variables and parameters are adopted throughout the two above-mentioned cases and some of them are used through the whole paper as well.

- w A constant that is inversely proportional to the computation speed of a processor. The processor can process the entire load in time wT_{cp}.
- Z A constant that is inversely proportional to the channel speed between two adjacent processors. The entire load can be transmitted over the channel in time ZT_{cm}.
- T_{cp} The time that it takes a processor to process the entire load when $w = 1$.
- T_{cm} The time it takes to transmit the entire load over a link where $Z = 1$. For simplicity, a constant bandwidth link is implicitly assumed.
- w_{eq}^{∞} The inverse speed of a single equivalent processor which is capable of replacing an infinite number of processors in the network and having the same performance as the original network.

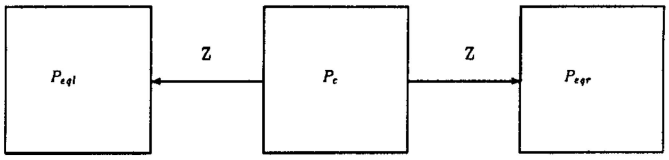


Fig. 2. Reduced daisy chain where load originates at network interior.

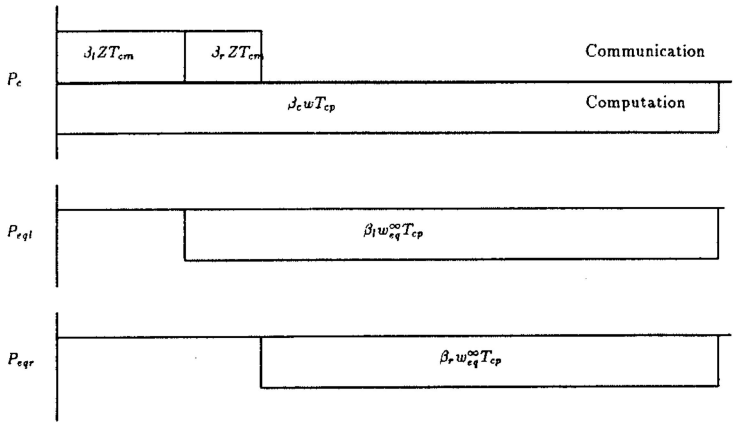


Fig. 3. Timing diagram for Fig. 2 system with front-end processors.

B. Linear Daisy Chain with Front-End Processors and Interior Load Origination

In this subsection, each processor in a linear daisy chain of processors is equipped with a front-end processor for communications off-loading. That is, the front-end processor relieves the main processor of communication duties so that it can concentrate on computation. Thus, each processor can compute and communicate at the same time.

Consider a linear daisy chain as in Fig. 1 but with an infinite number of processors. The load is now delivered to an interior processor and distributed to the other processors from this interior processor in a particular sequence without losing generality. We note that other sequences [24] could be considered. This interior processor first determines its fraction of the processing load β_c. The interior processor can immediately start processing this fraction. Simultaneously, it randomly selects either one of its immediate neighbors, say the left one, and transmits that processor's share of the load β_l to it in time $\beta_l ZT_{cm}$. Then it transmits the share of the immediate right neighbor processor β_r in time $\beta_r ZT_{cm}$. The left processor will share its portion of the total processing load β_l with an infinite number of processors to its left. Similarly, the right processor will share its portion of the total processing load β_r with an infinite number of processors to its right. Therefore, the left and right processors can each be viewed as a boundary processor in an infinite linear daisy chain network where the load originates at boundary. Thus the infinite chain of processors to the right and left of the central processor can each be replaced with a single processor with equivalent speed constants w_{eql}^{∞} and w_{eqr}^{∞}, respectively. Naturally, $w_{eqr}^{\infty} = w_{eql}^{\infty} = w_{eq}^{\infty}$.

A reduced linear daisy chain for the system shown in Fig. 1 is depicted in Fig. 2. There are three processors: the central processor P_c, an equivalent processor for an infinite number of processors to the left or "left equivalent processor" P_{eql}, and an equivalent processor for an infinite number of processors to the right or "right equivalent processor" P_{eqr}.

The timing diagram of the reduced daisy chain, shown in Fig. 2, is depicted in Fig. 3. The finish time is the time when all of the processors have finished computation. From the timing diagram it can be seen that the ultimate finish time limit T_{fei}^{∞} can be computed in three different ways. First, it equals the computing time of the central processor, $\beta_c wT_{cp}$. Second, it equals the communication time between the central processor and the left equivalent processor $\beta_l ZT_{cm}$ plus the computing time of the left equivalent processor $\beta_l w_{eq}^{\infty} T_{cp}$. Third, it equals the communication time between the central processor with the left equivalent processor and the right equivalent processor $(\beta_l + \beta_r)ZT_{cm}$ plus the computing time of the right processor. As described below, the three processors in Fig. 2 can be replaced with a single "equivalent" processor with equivalent speed constant w_{eqs}^{∞}. This equivalent processor is able to preserve the performance characteristics of the original system in Fig. 1. Then the time that it takes the system processor to compute the whole load, $w_{eqs}^{\infty} T_{cp}$, would equal T_{fei}^{∞}.

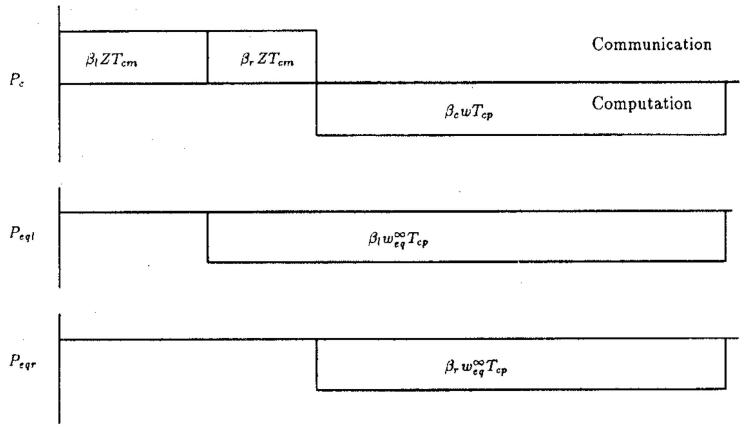


Fig. 4. Timing diagram for Fig. 2 system without front-end processors.

These four mentioned equations and the normalized sum of the fractions of the load equation are stated below:

$$T_{fei}^{\infty} = \beta_c w T_{cp} \tag{1}$$

$$T_{fei}^{\infty} = \beta_l Z T_{cm} + \beta_l w_{eq}^{\infty} T_{cp} \tag{2}$$

$$T_{fei}^{\infty} = (\beta_l + \beta_r) Z T_{cm} + \beta_r w_{eq}^{\infty} T_{cp} \tag{3}$$

$$T_{fei}^{\infty} = w_{eqs}^{\infty} T_{cp} \tag{4}$$

$$\beta_c + \beta_r + \beta_l = 1. \tag{5}$$

Using the above equations, where there are four unknowns, an explicit expression to calculate w_{eqs}^{∞} can be developed

$$w_{eqs}^{\infty} = \frac{w(ZT_{cm} + w_{eq}^{\infty} T_{cp})}{ZT_{cm} + w_{eq}^{\infty} T_{cp} + wT_{cp} + \dfrac{ww_{eq}^{\infty} T_{cp}^2}{ZT_{cm} + w_{eq}^{\infty} T_{cp}}}. \tag{6}$$

Here w_{eq}^{∞} is the speed of an equivalent processor for a linear daisy chain with an infinite number of processors, with front-end processors and with load origination at a boundary processor. It is given by the following equation from [14]:

$$w_{eq}^{\infty} = \frac{-Z\rho + \sqrt{(Z\rho)^2 + 4wZ\rho}}{2}. \tag{7}$$

Here $\rho = T_{cm}/T_{cp}$.

The ultimate finish time limit for load origination at the network interior, with front-end processors T_{fei}^{∞} can now simply be determined

$$T_{fei}^{\infty} = w_{eqs}^{\infty} T_{cp}. \tag{8}$$

C. Linear Daisy Chain with No Front-End Processors and Interior Load Origination

Consider a linear daisy chain similar to that mentioned in the previous section, except that the processors have no front-end processor. That is, the main processor, in addition to its responsibility for computation, must also handle communication duties. In other words, each processor in the daisy chain can either communicate or compute, but not do both at the same time. Again, all processors have the same computing speed constant w and all links have the same transmission speed constant Z.

One can follow a similar approach to that used in the front-end processor case where load originates at the network interior. One can collapse the original daisy chain with an infinite number of processors into three processors: a central processor, a "left equivalent processor", and a "right equivalent processor." A pictorial representation of the reduced daisy chain is shown in Fig. 2 and its timing diagram is shown in Fig. 4.

From the timing diagram it can be seen that the ultimate finish time limit T_{nfei}^{∞} can be computed in three different ways, in a manner similar to the case of Section IIB. The three processors in Fig. 2 can be replaced with a single equivalent processor with equivalent speed constant w_{eqs}^{∞} that is able to preserve the performance characteristics of the original system in Fig. 1. Then the time that it takes the system processor to compute the whole load $w_{eqs}^{\infty} T_{cp}$ would equal T_{nfei}^{∞}.

The above four mentioned equations and the normalized sum of the fractions of the load equation are stated below:

$$T_{nfei}^{\infty} = (\beta_l + \beta_r) Z T_{cm} + \beta_c w T_{cp} \tag{9}$$

$$T_{nfei}^{\infty} = \beta_l Z T_{cm} + \beta_l w_{eq}^{\infty} T_{cp} \tag{10}$$

$$T_{nfei}^{\infty} = (\beta_l + \beta_r) Z T_{cm} + \beta_r w_{eq}^{\infty} T_{cp} \tag{11}$$

$$T_{nfei}^{\infty} = w_{eqs}^{\infty} T_{cp} \tag{12}$$

$$\beta_c + \beta_r + \beta_l = 1. \tag{13}$$

Using the above equations, where there are four unknowns, an explicit expression to calculate w_{eqs}^{∞} can

be developed

$$w_{eqs}^{\infty} = \frac{w(ZT_{cm} + w_{eq}^{\infty}T_{cp})^2}{(w_{eq}^{\infty}T_{cp})^2 + w_{eq}^{\infty}T_{cp}^2 w + wT_{cp}(ZT_{cm} + w_{eq}^{\infty}T_{cp})}. \quad (14)$$

Here w_{eq}^{∞} is the speed of an equivalent processor of a linear daisy chain with an infinite number of processors, with no front-end processors and with load origination at a boundary processor. From [14] it is

$$w_{eq}^{\infty} = \sqrt{\rho Z w}. \quad (15)$$

Here $\rho = T_{cm}/T_{cp}$.

The ultimate finish time limit for load origination at the network interior, with no front-end processors, T_{nfei}^{∞} can now simply be determined:

$$T_{nfei}^{\infty} = w_{eqs}^{\infty}T_{cp}. \quad (16)$$

In closing this section it should be noted [1, 14] that there are certain combinations of link and processors speed parameters for linear daisy chains without front-end processors for which load distribution between processors is not optimal. That is, for these combinations faster solutions can be achieved when the load is run on a subset of processors.

III. TREE NETWORKS

A. Introduction

In this section load distribution for networks with a tree topology is discussed. Tree topologies to be considered are naturally connected and acyclic. This material is more general than a simple consideration of hard-wired tree networks of processors. This is because a natural way to distribute load in a processor network with cycles is through the use of an embedded spanning tree.

In this section a binary tree network of communicating processors is considered. The general technique developed here can be applied to other types of tree networks. In the tree there are three types of processors: root, intermediate, and terminal processors. Each tree has one root processor that originates the load. An intermediate processor can be viewed as a parent of lower level processors with which it has a direct connection. Also it is a child of an upper level processor with which it has a direct connection. The terminal processors can only be children processors.

Every processor can only communicate with its children processors and parent processor. Each of the processors in the tree is assumed to have the same computational speed, $1/w$. The communication speed between a parent processor and each of its children is also assumed to have the same value, $1/Z$.

In this section, two types of binary trees are discussed. One is where processors are equipped with front-end processors for communications off-loading. Therefore, communication and computation can take place in each processor at the same time. In the second type of tree, processors do not have front-end processors. That is, processors can either communicate or compute but not do both at the same time.

In [2] a finite tree for the above two cases was discussed. It was stated that the minimum processing time is achieved when all processors in the tree stop at the same time. The same intuition used in the linear daisy chain case can be used as a justification. Moreover formal proofs of optimality of single level trees are available [20, 21]. As the size of the tree gets larger, the share assigned to the root processor gets smaller and so the processing time decreases. On the other hand, adding more processors (nodes) to the tree, will result in more overhead time spent in communicating small fractions of load to the new processors. At some point, adding more processors will not decrease the fractions of load assigned to the root processor substantially and so there is not a considerable improvement in the processing time. In that case, it may be advisable not to add more processors (hardware) to the tree since the cost of doing so may not be worth the small improvement in the performance of the system.

To solve for the ultimate finish time limit, consider a binary tree with an infinite number of processors; that is, $n = \infty$ in Fig. 5. In the following the same definitions for T_{cp}, T_{cm}, w, and w_{eq}^{∞} are used as in the previous section; however, Z is defined as follows. Z is a constant that is inversely proportional to the channel speed between a parent processor and each of its children. The entire load can be transmitted over the channel in time ZT_{cm}. Again, for simplicity, the use of a constant bandwidth channel is assumed.

B. The Tree Network With Front-End Processors

The idea behind obtaining the processing time for this tree where $n = \infty$ is to collapse the tree into three processors as shown in Fig. 6. The right side of the tree has been replaced by one equivalent processor with equivalent processing speed w_{eq}^{∞}. The same is true for the left side of the tree where it was replaced with one equivalent processor that has an equivalent computational speed w_{eq}^{∞}. Naturally, as the left and right sub-trees are homogeneous infinite trees in their own right, an equivalent processor for either one of them has the same computational speed as one for the entire tree.

The timing diagram for this equivalent system, that preserves the characteristics of an infinite size binary tree, is shown in Fig. 7. From Fig. 7 it can be seen that the computing time of the root processor $\alpha_0 wT_{cp}$, equals the communication time between the parent processor (root processor) and the left processor $\alpha_l ZT_{cm}$, plus the computing time of the left equivalent processor $\alpha_l w_{eq}^{\infty}T_{cp}$. Also the computing time of the

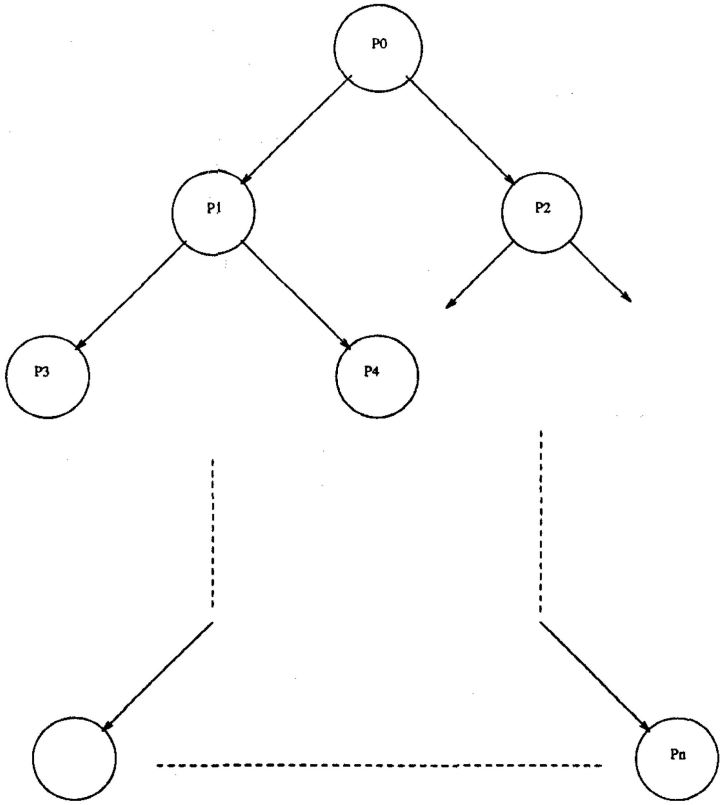


Fig. 5. Binary tree network.

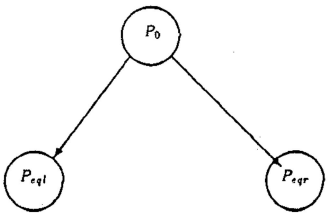


Fig. 6. Reduced tree network.

left side equivalent processor $\alpha_l w_{eq}^{\infty} T_{cp}$, equals the communication time between the root processor and the right side equivalent processor $\alpha_r Z T_{cm}$, plus the computing time of the right equivalent processor $\alpha_r w_{eq}^{\infty} T_{cp}$. If the three processors in Fig. 6 are replaced with one equivalent processor then the computing time of the root processor $\alpha_0 w T_{cp}$ equals the computing time of the equivalent processor $w_{eq}^{\infty} T_{cp}$. The three equations explained above are listed below:

$$\alpha_0 w T_{cp} = \alpha_l Z T_{cm} + \alpha_l w_{eq}^{\infty} T_{cp} \tag{17}$$

$$\alpha_l w_{eq}^{\infty} T_{cp} = \alpha_r Z T_{cm} + \alpha_r w_{eq}^{\infty} T_{cp} \tag{18}$$

$$\alpha_0 w T_{cp} = w_{eq}^{\infty} T_{cp}. \tag{19}$$

Also the sum of the fractions of the load equals one

$$\alpha_0 + \alpha_r + \alpha_l = 1. \tag{20}$$

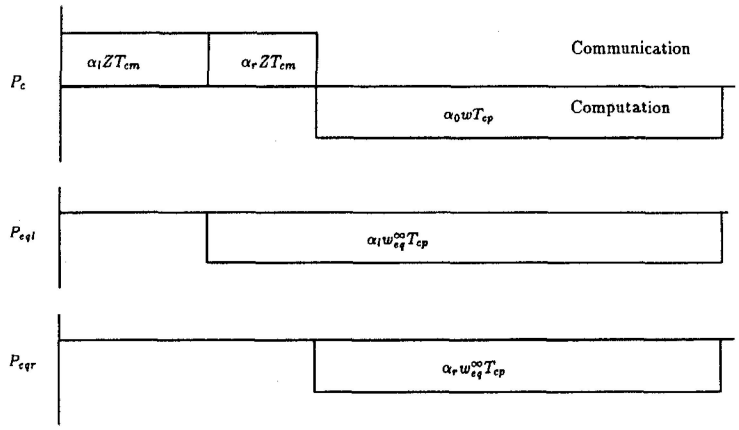


Fig. 7. Timing diagram for Fig. 6 system with front-end processors.

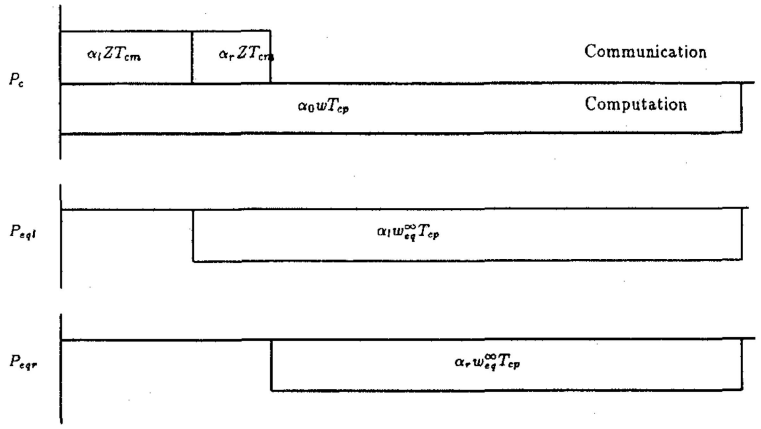


Fig. 8. Timing diagram for Fig. 6 system without front-end processors.

Now, there are four equations with four unknowns, namely w_{eq}^{∞}, α_0, α_r, and α_l. Thus w_{eq}^{∞} can be determined by solving iteratively the equation:

$$w_{eq}^{\infty} = \frac{w(ZT_{cm} + w_{eq}^{\infty} T_{cp})}{ZT_{cm} + w_{eq}^{\infty} T_{cp} + wT_{cp} + \dfrac{ww_{eq}^{\infty} T_{cp}^2}{ZT_{cm} + w_{eq}^{\infty} T_{cp}}}. \tag{21}$$

Alternately, the above iterative equation can be transformed through algebraic manipulation into a cubic equation:

$$(w_{eq}^{\infty})^3 + [2Z\rho + w](w_{eq}^{\infty})^2 - [Z\rho(w - Z\rho)]w_{eq}^{\infty} - wZ^2\rho^2 = 0. \tag{22}$$

Here $\rho = T_{cm}/T_{cp}$. The solution to such cubic equations appears in most mathematical handbooks [36].

Consequently, the ultimate finish time for an infinite tree network with front-end processors T_{fe}^{∞} can now be computed by

$$T_{fe}^{\infty} = w_{eq}^{\infty} T_{cp}. \tag{23}$$

C. The Tree Network Without Front-end Processors

Consider now the case where the processors in the tree network are not equipped with front-end processors. Therefore, each processor in the tree can either compute or communicate but not do both at the same time. The analytical results for a minimum finish time for a finite tree were considered in [2]. In this section, the situation where there is an infinite number of processors in a binary tree network is considered. As before, the left branch below the root as well as the right branch below the root processor are each collapsed into one equivalent processor. This equivalent processor is able to present the same characteristics as the original sub-tree. The timing of the reduced tree is depicted in Fig. 8 where α_0, α_r, α_l, Z, T_{cp}, T_{cm}, w_{eq}^{∞}, and w are defined as before. From Fig. 8 it can be seen the ultimate finish time limit with no front-end processor (nfe), T_{nfe}^{∞} can be computed in four different ways. This can be done in a manner similar to the previous case. The four equations are listed below:

$$T_{nfe}^{\infty} = (\alpha_l + \alpha_r)ZT_{cm} + \alpha_0 wT_{cp} \tag{24}$$

$$= \alpha_l ZT_{cm} + \alpha_l w_{eq}^{\infty} T_{cp} \tag{25}$$

$$= (\alpha_l + \alpha_r)ZT_{cm} + \alpha_r w_{eq}^{\infty} T_{cp} \tag{26}$$

$$= w_{eq}^{\infty} T_{cp}. \tag{27}$$

Also the total sum of the fractions of the load is equal to one:

$$\alpha_0 + \alpha_r + \alpha_l = 1. \tag{28}$$

Solving the above equations, one can find an expression for the exact numerical value of w_{eq}^{∞} by iteration

$$w_{eq}^{\infty} = \frac{w(ZT_{cm} + w_{eq}^{\infty} T_{cp})^2}{(w_{eq}^{\infty} T_{cp})^2 + 2ww_{eq}^{\infty} T_{cp}^2 + wZT_{cp}T_{cm}}. \tag{29}$$

Again, the above iterative equation can be transformed into a cubic equation through algebraic manipulation:

$$(w_{eq}^{\infty})^3 + w(w_{eq}^{\infty})^2 - [wZ\rho]w_{eq}^{\infty} - wZ^2\rho^2 = 0. \tag{30}$$

Here $\rho = T_{cm}/T_{cp}$.

The ultimate finish time limit can be computed using (27).

IV. PERFORMANCE EVALUATION

To examine the effect of the speed of the processors and the channel speed on the optimal finish time, the previous equations were used to obtain a set of curves, shown in Fig. 9. Both the performance of trees and linear daisy chains, with and without front-end processors, are plotted. For the linear daisy chain, load origination at both the network interior and, as a reference, the boundary is plotted. Here the

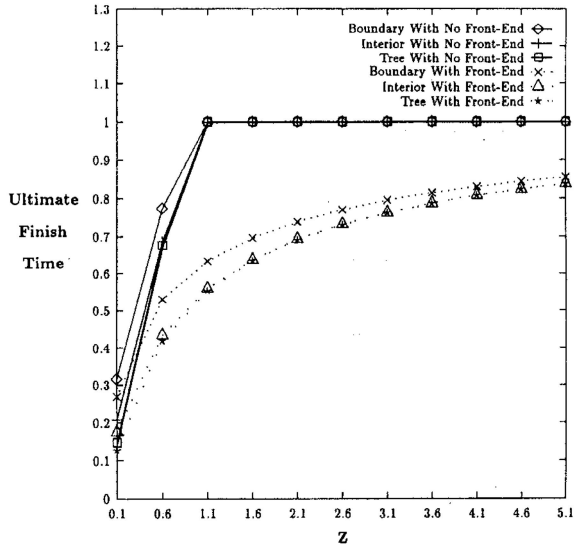


Fig. 9. Ultimate finish time for linear daisy chain with load origination at network boundary and interior and with and without front-end processors and for tree network with and without front-end processors. $T_{cm} = T_{cp} = 1$, $w = 1$.

ultimate minimum finish time is plotted against Z. Also, $T_{cm} = 1$, $T_{cp} = 1$, and $w = 1$. These parameter values are chosen to be representative.

1) Fig. 9 supports the intuition that the ultimate minimum finish time increases as the channel speed decreases. A similar increase was observed when processor speed decreases.

2) For certain cases, it has been observed that beyond a threshold value of Z the overhead of communication time that is needed to distribute the load to the rest of the processors in the network becomes excessive. Thus using a single processor to execute the whole job would be more efficient. For the chosen parameters a single processor takes wT_{cp} to compute the whole load. Therefore, a single processor is selected whenever the ultimate finish time exceeds wT_{cp}. This accounts for the horizontal straight lines in the figure.

3) There are two, largely intuitive, trends apparent in the curves. First, the use of the front-end processors improves the minimum finish time. Secondly, for the linear daisy chain, origination at the interior of the chain is superior to the origination at the boundary.

4) One surprising result in Fig. 9 is that, at least for some parameter values, a binary tree network is only marginally faster than a linear daisy chain with origination at the chain interior. This can be partially explained by noting that in both cases the load is distributed for the first three processors in an identical fashion. The majority of the load is allocated to the first three processors when link speeds are moderate to slow. For this range the additional processors in the lower level of the tree do not lead to a significant performance improvement.

V. CONCLUSION

This is an exciting problem area as one can demonstrate a fundamental limit of performance in a problem involving communication/computation tradeoffs in a relatively straightforward manner. The importance of these results is that they quantify the fact that a finite-sized processor network load sharing a divisible job can perform almost as well as an infinite sized network. This is good practical news and allows such infinite-sized networks to be used as a theoretical performance benchmark.

REFERENCES

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

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

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

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

[5] Baumgartner, K. M., and Wah, B. W. (1989)
GAMMON: A load balancing strategy for local computer systems with multiaccess networks.
IEEE Transactions on Computers, **38** (Aug. 1989), 1098–1109.

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

[7] Lo, V. M. (1988)
Heuristic algorithms for task assignment in distributed systems.
IEEE Transactions on Computers, **37** (Nov. 1988), 1384–1397.

[8] Ramamrithamm, K., Stankovic, J. A., and Zhao, W. (1989)
Distributed scheduling of tasks with deadlines and resources requirements.
IEEE Transactions on Computers, **38** (Aug. 1989), 1110–1122.

[9] Shin, K. G., and Chang, Y-C. (1989)
Load sharing in distributed real-time systems with state change broadcasts.
IEEE Transactions on Computers, **38** (Aug. 1989), 1124–1142.

[10] Stone, H. S. (1977)
Multiprocessor scheduling with the aid of network flow algorithms.
IEEE Transaction on Software Engineering, **SE-3** (Jan. 1977), 85–93.

[11] Mirchandaney, R., Towsley, D., and Stankovic, J. A. (1989)
Analysis of the effects of delays on the load sharing.
IEEE Transactions on Computers, **38** (Nov. 1989), 1513–1525.

[12] Ni, L. M., and Hwang, K. (1985)
Optimal load balancing in a multiple processor system with many job classes.
IEEE Transaction on Software Engineering, **SE-11** (May 1985), 491–496.

[13] Bataineh, S., and Robertazzi, T. G. (1992)
Ultimate performance limit for networks of load sharing processors.
In *Proceedings of the 1992 Conference on Information Sciences and Systems*, Princeton University, Princeton, NJ, Mar. 1992, 794–799.

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

[15] Zemanian, A. H. (1988)
Infinite electric networks: A reprise.
IEEE Transactions on Circuits and Systems, **35** (Nov. 1988), 1346–1358.

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

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

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

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

[20] Bharadwaj, V., Ghose, D., and Mani, V. (1992)
A study of optimality conditions for load distribution in tree networks with communication delay.
Technical report 423/GI/02-92, Guidance and Instrumentation Laboratory, Dept. of Aerospace Engineering, Indian Institute of Science, Bangalore, India.

[21] Sohn, J., and Robertazzi, T. G. (1993)
Optimal load sharing for a divisible job on a bus network.
In *Proceedings of the 1993 Conference on Information Sciences and Systems*, The Johns Hopkins University, Baltimore, MD, Mar. 1993, 835–840; also in *IEEE Transactions on Aerospace and Electronic Systems*, **32** (Jan. 1996), 34–40.

[22] Bataineh, S., Hsiung, T., and Robertazzi, T. G. (1994)
Closed-form solutions for bus and tree networks of processors load sharing a divisible job.
Presented at the 1993 International Conference on Parallel Processing, Chicago, IL, Aug. 1993; also in *IEEE Transactions on Computers*, **43** (Oct. 1994), 1184–1196.

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

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

[25] Du, J., and Leung, J. Y.-T. (1989)
Complexity of scheduling parallel task systems.
SIAM Journal on Discrete Mathematics, **2** (Nov. 1989), 473–487.

[26] Blazewicz, J., Drabowski, M., and Weglarz, J. (1986)
Scheduling multiprocessor tasks to minimize schedule length.
IEEE Transactions on Computers, **C-35** (May 1986), 389–393.

[27] Zhao, W., Ramamritham, K., and Stankovic, J. A. (1987)
Preemptive scheduling under time and resource constraints.
IEEE Transactions on Computers, **C-36** (Aug. 1987), 949–960.

[28] Ramamritham, K., Stankovic, J. A., and Shiah, P.-F. (1990)
Efficient scheduling algorithms for real-time multiprocessor systems.
IEEE Transactions on Parallel and Distributed Systems, **1** (Apr. 1990), 184–194.

[29] Lee, C.-H., Lee, D., and Kim, M. (1992)
Optimal task assignment in linear array networks.
IEEE Transactions on Computers, **41** (July 1992), 877–880.

[30] Peng, D.-T., and Shin, K. G. (1993)
A new performance measure for scheduling independent real-time tasks.
Journal of Parallel and Distributed Computing, **19** (1993), 11–26.

[31] Xu, J., and Hwang, K. (1993)
Heuristic methods for dynamic load balancing in a message-passing multicomputer.
Journal of Parallel and Distributed Computing, **18** (1993), 1–13.

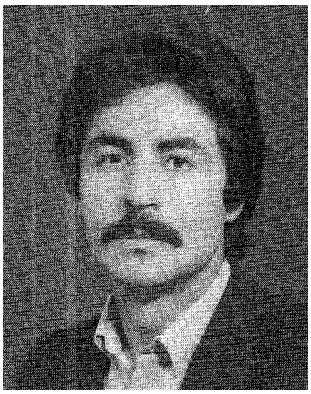
[32] Ahmad, I., Ghafoor, A., and Fox, G. C. (1994)
Hierarchical scheduling of dynamic parallel computations on hypercube multicomputers.
Journal of Parallel and Distributed Computing, **20** (1994), 317–329.

[33] Blazewicz, J., and Drozdowski, M. (1995)
Scheduling divisible jobs on hypercubes.
Parallel Computing, **21** (1995), 1945–1956.

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

[35] Haddad, E. (1994)
Communication protocol for optimal redistribution of divisible load in distributed real-time systems.
In *Proceedings of the ISMM International Conference on Intelligent Information Management Systems*, Washington, DC, June 1994, 39–42.

[36] Burington, R. S. (1973)
Handbook of Mathematical Tables and Formulas (5th ed.).
New York: McGraw-Hill, 1973.



Sameer Bataineh received the B.S. degree in electrical engineering in 1985 from Syracuse University, Syracuse, NY. He received the Ph.D. degree in electrical engineering from the University at Stony Brook, Stony Brook, NY, in 1992.

Since 1992 he has been on the faculty of the Electrical Engineering Dept. of the Jordan University of Science and Technology, Irbid, Jordan.

Thomas G. Robertazzi (S'75—M'77—S'78—M'81—SM'91) received the B.E.E. from Cooper Union in 1977 and the Ph.D in electrical engineering from Princeton University, Princeton, NJ, in l981.

He was an Assistant Professor of Electrical Engineering at Manhattan College, Riverdale, NY, during 1982–1983. Since 1983 he has been at the University at Stony Brook where he is presently an Associate Professor in the Electrical Engineering Department. During the Fall of 1990, he was a visiting research scientist at Columbia University's Electrical Engineering Department. His research interests are in the performance evaluation of computer networks and computer systems.

Dr. Robertazzi has been editor-in-chief for books for the Communications Society and an associate editor of *Wireless Networks*.