STATE UNIVERSITY OF NEW YORK AT

STONY BROOK

CEAS TECHNICAL REPORT 623

Ultimate Performance Limits for Networks of Load Sharing Processors

S. Bataineh & T.G. Robertazzi

April 2, 1992

Ultimate Performance Limits for Networks of Load Sharing Processors

Sameer Bataineh and Thomas Robertazzi

Dept. of Electrical Engineering,

SUNY at Stony Brook,

Stony Brook, N.Y 11794

(516)-632-8412/8400

ABSTRACT

Ultimate performance limits to the aggregate processing speed of networks of load sharing processors 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 load sharing processors inter-connected in either a linear daisy chain or tree network. Such limits are important as they provide an ideal baseline to compare the performance of finite configurations of processors against.

Index Terms:

Load Sharing, Load Balancing, Divisible Job, Multiprocessors, Scheduling Algorithm.

1 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, 6, 7, 8, 9, 10, 11, 12]. This previous work involved the paradigm of indivisible jobs. Under this paradigm a job can be processed by at most one processor. A new paradigm of divisible jobs was discussed in [1, 2, 3, 4] in which the authors examined the case where a job can be split 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 and image processing. The optimal size of the fractions that have to be allocated to each processor in order to achieve the minimum finish time were calculated for linear daisy chains of processors[1], tree network of processors[2], and processors interconnected thru a bus[3, 4].

In this work it was noticed that the system speed followed a "law of diminishing returns". That is, as more processors are added to the load sharing network, the solution time levels off and saturates. But the question of the ultimate performance limit, which is obtained when an infinite number of processors are used, was not discussed.

In this paper we consider the case of performance for an infinite number of processors. This, of course, make it possible to achieve the ultimate performance limit in terms of the number of processors for a specific interconnection topology. Knowing the ultimate processing time, one simply 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.

The paper is organized as follows: The second section discusses the linear daisy chain network where load is either originated at a boundary processor or at a processor in the interior of the network. Systems with and without front-end processors are analyzed. The

third section discusses tree networks with and without front-end processors. Section four presents performance evaluation results. The conclusion appears in section five.

2 Linear Daisy Chain

2.1 Introduction

Consider a linear daisy chain of processors as in Fig. 1. Each processor has the same computational speed, $\frac{1}{w}$, and the channel speed between two adjacent processors is $\frac{1}{Z}$. 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 solution time thru parallel processing. 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]. Although the minimum finish time was found to decrease as the number of processor increases, it was also found that after a certain number of processors 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 worth the small improvement in performance.

The ultimate minimum finish time limit, T^{∞}, will be achieved if there are an infinite number of processors in the chain. Such a limit is practically important as it provides an ideal baseline to compare the performance of a finite linear daisy chain of processors against. To solve for an ultimate finish time limit, we consider a daisy chain network with an infinite number of processors. Two cases, linear daisy chains with and without front-end processors, will be discussed.

The following definitions for some variables and parameters will be adopted through out the two above mentioned cases and some of them will be used through the whole paper as well:

T_{cp} : The time that it takes the 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 $Z = 1$.

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} .

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.

2.2 Linear Daisy Chain with Front−End Processors

In this subsection, each processor in the linear daisy chain is equipped with a front-end processor. That is, each processor can compute and communicate at the same time. We consider an infinite number of processors in the linear daisy chain . In other words, n in Fig. 1 equals infinity. The load may originate from a boundary processor or from a processor in the interior.

Load Origination at Network Boundary

Suppose 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 left processor starts working on its fraction of the load, α_1, in time $\alpha_1 w T_{cp}$. It also simultaneously communicates the remaining fraction of the load to processor 2 in time $(\alpha_2 + \alpha_3 \ldots + \alpha_\infty) Z T_{cm}$. Processor 2 can then begin computing its share of the load in time $\alpha_2 w_2 T_{cp}$ and communicating the rest of the load to processor 3 in time $(\alpha_3 + \alpha_4 \ldots + \alpha_\infty) Z T_{cm}$. The process will continue till each processor receives its share of the load and is working on the problem.

Two adjacent processors may be combined into a single "equivalent" processor that presents operating characteristics to the rest of the network that are identical to those of original two processors. This, basically, is the idea that enables us to collapse a linear daisy chain network, consisting of an infinite number of processors, into one "equivalent" processor that presents operating characteristics that are identical to those of the original system.

Consider first a finite chain of n processors. Let us start with the $n-1$st and the nth processors, as illustrated in Fig. 2. The figure begins at the moment when load has finished being transmitted to the $n-1$st processor from $n-2$nd processor. The $n-1$st processor keeps $\hat{\alpha}_{n-1}$ fraction of what it receives and transmits the remaining $1-\hat{\alpha}_{n-1}$ fraction to the n^{th} processor. The total load received by the $n-1$st processor from $n-2$nd processor is $\alpha_{n-1} + \alpha_n$. The time that each processor is active, from Fig. 3, is:

$$T_{n-1} = \hat{\alpha}_{n-1}(\alpha_{n-1} + \alpha_n) w T_{cp} \tag{2.1}$$

$$T_n = (1 - \hat{\alpha}_{n-1})(\alpha_{n-1} + \alpha_n)(Z T_{cm} + w T_{cp}) \tag{2.2}$$

The optimal choice for $\hat{\alpha}_{n-1}$, given in the following equation, is that which reflects the fact that both processors stop at the same time, since this will intuitively achieve the minimum finish time. Throughout this paper it will be assumed that all processors stop computing at the same time in order to achieve an optimal solution [1, 2]:

$$\hat{\alpha}_{n-1} = \frac{ZT_{cm} + wT_{cp}}{ZT_{cm} + 2wT_{cp}} \tag{2.3}$$

The two processors can be replaced by a single processor that is able to compute the load of processor n-1 and n, $(\alpha_{n-1} + \alpha_n)$, in time $(\alpha_{n-1} + \alpha_n)w_{eq}^{2p}T_{cp}$. This time should equal T_n and T_{n-1} in order to preserve the performance characteristics of the system shown in Fig. 2. Here w_{eq}^{2p} is the equivalent speed constant of the single processor and is given by:

$$w_{eq}^{2p} = \frac{\hat{\alpha}_{n-1}(\alpha_{n-1} + \alpha_n)wT_{cp}}{(\alpha_{n-1} + \alpha_n)T_{cp}} \tag{2.4}$$

$$= \hat{\alpha}_{n-1}w \tag{2.5}$$

Here $\hat{\alpha}_{n-1}$ is given by (2.3). Thus, starting with the last two processors in the chain, the entire linear daisy chain of processors can be collapsed, two at a time, into a single equivalent processor.

To find the equivalent speed constant, w_{eq}^{∞}, for an infinite number of processors in the daisy chain, let the network, shown in Fig. 4, consists of P_1 and an equivalent processor for the infinite chain consisting of $i = 2, 3, \ldots, \infty$. Naturally, this equivalent processor should have the same speed constant, w_{eq}^{∞}, as that for the infinite chain of processors $i = 1, 2, 3, \ldots, \infty$. It should be noted that this solution approach is used in infinite electric circuit theory to compute equivalent resistance.

$$w_{eq}^{\infty} = \hat{\alpha_1}w \tag{2.6}$$

If we substitute the value of $\hat{\alpha_1}$ from equation 2.3:

$$w_{eq}^{\infty} = \frac{Z\rho + w_{eq}^{\infty}}{Z\rho + 2w_{eq}^{\infty}}w \tag{2.7}$$

Here $\rho = \frac{T_{cm}}{T_{cp}}$ and solving for w_{eq}^{∞} results in :

$$w_{eq}^{\infty} = \frac{-Z\rho + \sqrt{(Z\rho)^2 + 4wZ\rho}}{2} \tag{2.8}$$

The ultimate finish time limit for a linear daisy chain with front-end processor and where originates at the network boundary is given by:

$$T_{feb}^{\infty} = w_{eq}^{\infty} T_{cp} \tag{2.9}$$

Load Origination at Network Interior

Consider a linear daisy chain similar to that in Fig. 1. The load is now delivered to an interior processor. Then this central processor should first determine its fraction of the processing load, β_c, and can immediately start working on it. Simultaneously, it should randomly select either one of its immediate neighbors, say the left one, and transmit its share of the load, β_l in time $\beta_l Z T_{cm}$. Then it should transmit the share of the immediate right neighbor processor, β_r, in time $\beta_r Z T_{cm}$. The left processor will share its share of the total processing load, β_l, with an infinite number of processors to its left. Similarly, the right processor will share its share 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 be replaced with 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. 5. 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. 5, is depicted in Fig. 6. 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, $\beta_c w T_{cp}$. Second, it equals the communication time between the central processor and the left equivalent processor, $\beta_l Z T_{cm}$, 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 , $(\beta_l+\beta_r) Z T_{cm}$, plus the computing time of the right processor. Finally, the three processors in Fig. 5 can be replaced with a single "equivalent" processor or "system 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 $w_{eqs}^{\infty} T_{cp}$, equals T_{fei}^{∞} .

In addition to the above four mentioned equations, the normalized sum of the fractions of the load equation are stated below:

$$T_{fei}^{\infty} = \beta_c w T_{cp} \tag{2.10}$$

$$T_{fei}^{\infty} = \beta_l Z T_{cm} + \beta_l w_{eq}^{\infty} T_{cp} \tag{2.11}$$

$$T_{fei}^{\infty} = (\beta_l + \beta_r) Z T_{cm} + \beta_r w_{eq}^{\infty} T_{cp} \tag{2.12}$$

$$T_{fei}^{\infty} = w_{eqs}^{\infty} T_{cp} \tag{2.13}$$

$$\beta_c + \beta_r + \beta_l = 1 \tag{2.14}$$

Using the above equations, where there are four unknowns, an expression to calculate

w_{eqs}^{∞} can be developed and is given below:

$$w_{eqs}^{\infty} = \frac{w(ZT_{cm} + w_{eq}^{\infty} T_{cp})}{ZT_{cm} + w_{eq}^{\infty} T_{cp} + wT_{cp} + \frac{ww_{eq}^{\infty} T_{cp}^2}{ZT_{cm} + w_{eq}^{\infty} T_{cp}}} \tag{2.15}$$

We assume that all processors have the same speed, $\frac{1}{w}$, and also all the channels between adjacent processors in the chain have the same speed, $\frac{1}{Z}$. Here w_{eq}^{∞} is given by equation (2.8).

The ultimate minimum finish time limit can now simply determined:

$$T_{fei}^{\infty} = w_{eqs}^{\infty} T_{cp} \tag{2.16}$$

2.3 Linear Daisy Chain with No Front–End Processors

Consider a linear daisy chain similar to that mentioned in the previous section, except for the fact that the processors have no front-end processor. In other words, each processor in the daisy chain can either communicate or compute, but not do both at the same time. As before there are two cases: either the load originates at the boundary processor or at an interior processor of the chain.

Load Origination at Network Boundary

Let the left most processor in the daisy chain, processor 1 in Fig. 1, receive the entire load that is to be shared with an infinite number of processors. Unlike the previous subsection where load originates at network boundary, the left processor, in this case has to first calculate the fractions of the load that has to be assigned to each processor in the chain and deliver it in time $(\alpha_2 + \alpha_3 + \ldots + \alpha_{\infty})ZT_{cm}$, and then compute its share of the load in time $\alpha_1 wT_{cp}$. So the ultimate minimum finish time limit, T_{nfeb}^{∞}, equals the sum of the communication and computing time of the left processor:

$$T_{nfeb}^{\infty} = (\alpha_2 + \alpha_3 + \ldots + \alpha_{\infty})ZT_{cm} + \alpha_1 wT_{cp} \tag{2.17}$$

As before, two adjacent processors may be combined into a single equivalent processor that presents the operating characteristics to the rest of the network that are identical to those of the original two processors. Starting with the last two processors in the chain, the nth and the $n-1$st processors, and using a similar technique as in the previous subsection, we can find an optimal value for $\hat{\alpha}_{n-1}$ that will minimize the finish time. This value is given below [1]:

$$\hat{\alpha}_{n-1} = \frac{w}{w+w} = \frac{1}{2} \tag{2.18}$$

The equivalent speed constant, w_{eq}^{2p}, of the single "equivalent" processor that replaced the nth and $n-1$st processors is given by

$$w_{eq}^{2p} = (1 - \hat{\alpha}_{n-1})(Z\rho + w) \tag{2.19}$$

where $\rho = \frac{T_{cm}}{T_{cp}}$.

Once again it is apparent that an equivalent processor for processors $1, 2, 3, \ldots, \infty$ has the same speed constant, w_{eq}^{∞}, as that of an equivalent processor for processors $2, 3, 4, \ldots, \infty$. These can be combined into an implicit equation along with the speed constant for processor 1 and can be solved as:

$$w_{eq}^{\infty} = \sqrt{\rho Z w} \tag{2.20}$$

The ultimate minimum finish time is :

$$T_{nfeb}^{\infty} = w_{eq}^{\infty} T_{cp} \tag{2.21}$$

Load Origination at Network Interior

Consider the daisy chain network shown in Fig. 1 where now each processor in the chain does not have a front-end processor and $n = \infty$.

Following a similar approach to that is used in the front-end processor case where load originates at the network interior, we 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. 5 and its timing diagram is shown in Fig. 7.

From the timing diagram it can be seen that the ultimate finish time limit, T^{∞}_{nfei}, can be computed in three different ways: first, it equals the communication time between the central processor and the left equivalent processor, $\beta_l Z T_{cm}$, plus the communication time between the central processor and the right equivalent processor, $\beta_r Z T_{cm}$, in addition to the computing time of the central processor, $\beta_c w T_{cp}$. Secondly, it equals the communication time between the central processor and the left equivalent processor, $\beta_l Z T_{cm}$, plus the computing time of the left equivalent processor, $\beta_l w^{\infty}_{eq} T_{cp}$. Third, it equals the the central processor's communication time with the left equivalent processor and the right equivalent processor, $(\beta_l + \beta_r) Z T_{cm}$, plus the computing time of the right processor, $\beta_r w^{\infty}_{eq} T_{cp}$. Finally, the three processors in Fig. 5 can be replaced with a single equivalent processor or "system 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 $w^{\infty}_{eqs} T_{cp}$, equals T^{∞}_{nfei} .

In addition to the above four mentioned equations, the normalized sum of the fractions of the load equation is stated below:

$$T^{\infty}_{nfei} = (\beta_l + \beta_r) Z T_{cm} + \beta_c w T_{cp} \tag{2.22}$$

$$T^{\infty}_{nfei} = \beta_l Z T_{cm} + \beta_l w^{\infty}_{eq} T_{cp} \tag{2.23}$$

$$T^{\infty}_{nfei} = (\beta_l + \beta_r) Z T_{cm} + \beta_r w^{\infty}_{eq} T_{cp} \tag{2.24}$$

$$T^{\infty}_{nfei} = w^{\infty}_{eqs} T_{cp} \tag{2.25}$$

$$\beta_c + \beta_r + \beta_l = 1 \tag{2.26}$$

Using the above equations, where there are four unknowns, an expression to calculate w^{∞}_{eqs} can be developed and is given below:

$$w^{\infty}_{eqs} = \frac{w(ZT_{cm} + w^{\infty}_{eq} T_{cp})^2}{(w^{\infty}_{eq} T_{cp})^2 + w^{\infty}_{eq} T^2_{cp} w + wT_{cp}(ZT_{cm} + w^{\infty}_{eq} T_{cp})} \tag{2.27}$$

Here w^{∞}_{eq} is given by equation (2.20).

The ultimate minimum finish time limit can now simply determined:

$$T^{\infty}_{nfei} = w^{\infty}_{eqs} T_{cp} \tag{2.28}$$

In closing this section it should be noted [1] 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.

3 Tree Network

3.1 Introduction

Consider a binary tree network of communicating processors. In the tree we have three types of nodes (processors): root, intermediate and terminal nodes. Each tree has one root node that originates the load. An intermediate node can be viewed as a parent of lower level nodes with which it has a direct connection. Also it is a child of an upper level node with which it has a direct connection. The terminal nodes can only be children nodes.

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

In this section, we will discuss two types of binary trees. One is where processors are equipped with front-end processors. 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 suggested that the minimum processing time is achieved when all processors in the tree stop at the same time and it was also pointed out that the total processing time is equivalent to the computation time of the root processor. The computational time of the root processor is proportional to the fraction of the load that has been assigned to it. As the size of the tree gets larger, the share of 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, we consider a binary tree with an infinite number of processors; that is, $n = \infty$ in Fig. 8. In the following we will use the same definitions for T_{cp}, T_{cm}, w, and w_{eq}^{∞} as in the previous section; however, Z is defined as follows:

Z : A constant that is inversely proportional to the channel speed between a parent processor and it's children. The entire load can be transmitted over the channel in time ZT_{cm}

3.2 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. 9. 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 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. 10. From Fig. 10 it can be seen that the computing time of the root processor, $\alpha_0 w T_{cp}$, equals the communication time between the parent processor (root processor) and the left processor, $\alpha_l Z T_{cm}$, plus the computing time of the left equivalent processor, $\alpha_l w_{eq}^{\infty} T_{cp}$. Also the computing time of the 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 we replace the three processors in Fig. 9 with one equivalent processor, called the "system processor" that has the power to compute the whole load while preserving the characteristics, of the original system, then the computing time of the root processor, $\alpha_0 w T_{cp}$, equals the computing time of the system processor, $w_{eq}^{\infty} T_{cp}$. The three equations explained above are listed below:

$$\alpha_0 w T_{cp} = \alpha_l Z T_{cm} + \alpha_l w_{eq}^{\infty} T_{cp} \tag{3.1}$$

$$\alpha_l w_{eq}^{\infty} T_{cp} = \alpha_r Z T_{cm} + \alpha_r w_{eq}^{\infty} T_{cp} \tag{3.2}$$

$$\alpha_0 w T_{cp} = w_{eq}^{\infty} T_{cp} \tag{3.3}$$

Also the sum of the fractions of the load equals one

$$\alpha_0 + \alpha_r + \alpha_l = 1 \tag{3.4}$$

Now, there are four equations with four unknowns, namely $w_{eq}^{\infty}, \alpha_0\, \alpha_r$,and α_l. Thus w_{eq}^{∞} can be determined by solving iteratively the following equation

$$w_{eq}^{\infty} = \frac{w(ZT_{cm} + w_{eq}^{\infty} T_{cp})}{ZT_{cm} + w_{eq}^{\infty} T_{cp} + wT_{cp} + \frac{w w_{eq}^{\infty} T_{cp}^2}{ZT_{cm} + w_{eq}^{\infty} T_{cp}}} \tag{3.5}$$

Consequently, the minimum finish time for an infinite tree network with front-end

processors, T_{fe}^{∞}, can now be computed by the following equation:

$$T_{fe}^{\infty} = w_{eq}^{\infty} T_{cp} \tag{3.6}$$

3.3 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 paper, we will consider the case where there is an infinite number of processors in a binary tree network and find the ultimate finish time limit. As before, the left branch as well as the right branch below the root processor is each collapsed into one equivalent processor that is able to present the same characteristics as the original sub-tree. The timing of the reduced tree is depicted in Fig. 11 where $\alpha_0, \alpha_r, \alpha_l, Z, T_{cp}, T_{cm}, w_{eq}^{\infty}$ and w_s are defined as before. From Fig. 11 it can be seen the ultimate finish time limit, T_{nfe}^{∞} can be computed in four different ways: first, it is equal to the communication time between the root processor with the left and the right processors, $(\alpha_l + \alpha_r)ZT_{cm}$, plus the computing time of the root processor, $\alpha_0 wT_{cp}$. Secondly, it equals the sum of the communication time between the root processor and the left processor, $\alpha_l ZT_{cm}$, and the computing time of the left processor, $\alpha_l w_{eq}^{\infty} T_{cp}$. Third, it equals the communication time between the root with the left and the right processors, $(\alpha_l + \alpha_r)ZT_{cm}$, and the computing time of the right processor, $\alpha_r w_{eq}^{\infty} T_{cp}$. Fourth, it equals $w_{eq}^{\infty} T_{cp}$. The four equations are listed below:

$$T_{nfe}^{\infty} = (\alpha_l + \alpha_r)ZT_{cm} + \alpha_0 wT_{cp} \tag{3.7}$$

$$= \alpha_l ZT_{cm} + \alpha_l w_{eq}^{\infty} T_{cp} \tag{3.8}$$

$$= (\alpha_l + \alpha_r) Z T_{cm} + \alpha_r w_{eq}^{\infty} T_{cp} \tag{3.9}$$

$$= w_{eq}^{\infty} T_{cp} \tag{3.10}$$

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

$$\alpha_0 + \alpha_r + \alpha_l = 1 \tag{3.11}$$

Solving the above system equations, we can find an expression that enables us to determine the exact numerical value of w_{eq}^{∞} by iteration.

$$w_{eq}^{\infty} = \frac{w(ZT_{cm} + w_{eq}^{\infty} T_{cp})^2}{(w_{eq}^{\infty} T_{cp})^2 + w w_{eq}^{\infty} T_{cp}^2 + w T_{cp}(ZT_{cm} + w_{eq}^{\infty} T_{cp})} \tag{3.12}$$

The ultimate finish time limit can be computed using equation (3.12).

4 Performance Evaluation

To examine the effect of the speed of the processors and the channel speed on the ultimate minimum finish time, two sets of plots were obtained . In the first the ultimate minimum finish time is plotted against Z and in the second the ultimate minimum finish time is plotted against w. In both sets $T_{cm} = 1$ and $T_{cp} = 1$. In the first set $w = 1$ while in the second $Z = 1$.

- Figures, 12... 18, support the intuition that the ultimate minimum finish time increases as the the speed of the processors, the channel speed or both decreases.

- 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 so that 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 explains the straight lines in the figures.

- There are two, largely intuitive, trends apparent in the figures. 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.

- One surprising result appears on Fig. 16, Fig. 17 and Fig. 18 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. Since 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.

5 Conclusion

This is an exciting problem area as one can demonstrate a fundamental limit of performance in a problem involving communication/computation tradeoffs in relatively straightforward manner. The fact that this parallel computing network and network problem can be solved using a well known circuit theory technique is a secondary point of interest.

Acknowledgements

The research in this paper is supported by the SDIO/IST and managed by the office of the Naval Research under grant No. N00014-91-J4063.

References

[1] Cheng,Y.C. and Robertazzi, T.G., "Distributed Computation with Communication Delays", *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 24, No. 6, Nov. 1988, pp. 700-712.

[2] Cheng, Y.C. and Robertazzi, T.G., "Distributed Computation for Tree Network with Communication Delays", *IEEE Transactions on Aerospace and Systems*, Vol. 26, No. 3, May 1990, pp. 511-516.

[3] Bataineh, S. and Robertazzi, T.G., "Distributed Computation for a Bus Networks with Communication Delays", *Proceedings of the 1991 Conference on Information Sciences and Systems*, The Johns Hopkins University, Baltimore MD, March 1991, pp. 709-714.

[4] Bataineh, S. and Robertazzi, T.G., "Bus oriented Load Sharing for a Network of Sensor Driven Processors", *IEEE Transactions on Systems, Man and Cybernetics*, Sept. 1991, Vol.21, No. 5.

[5] Baumgartner, K.M. and Wah, B.W., "GAMMON: A Load Balancing Strategy for Local Computer Systems with Multiaccess Networks", *IEEE Transactions on Computers*, Vol. 38, No. 8, August 1989, pp. 1098-1109.

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

[7] Lo, V.M., "Heuristic Algorithms for Task Assignment in Distributed Systems", *IEEE Transactions on Computers*, Vol. 37, No.11, Nov. 1988, pp. 1384-1397.

[8] Ramamrithamm K., Stankovic, J.A. and Zhao, W., "Distributed Scheduling of Tasks with Deadlines and Resources Requirements", *IEEE Transactions on Computers*, Vol. 38, No. 8, August 1989, pp. 1110-1122.

[9] Shin, K.G. and Chang, Y-C., " Load Sharing in Distributed Real-Time Systems with State Change Broadcasts", *IEEE Transaction on Computers*, Vol. 38, No. 8, August 1989, pp. 1124-1142.

[10] Stone, H.S., "Multiprocessor Scheduling with the Aid of Network Flow Algorithms", *IEEE Transaction on Software Engineering*, Vol . SE-3, No. 1, Jan. 1977, pp. 85-93.

[11] Mirchandaney, R. Towsley, D. and Stankovic, J.A., "Analysis of the Effects of Delays on the Load Sharing", *IEEE Transactions on Computers*, Vol. 38, No. 11, Nov. 1989, pp. 1513-1525.

[12] Ni, L.M. and Hwang, K., "Optimal Load Balancing in a Multiple Processor System with Many Job Classes", *IEEE Transaction on Software Engineering*, Vol. SE-11, No. 5, May 1985, pp. 491-496.

[13] Bataineh, S. and Robertazzi, T.G., "Ultimate Performance Limit for Networks of Load Sharing Processors", Technical Report, March 1992, Available from T. Robertazzi.

Daisy Chain Network

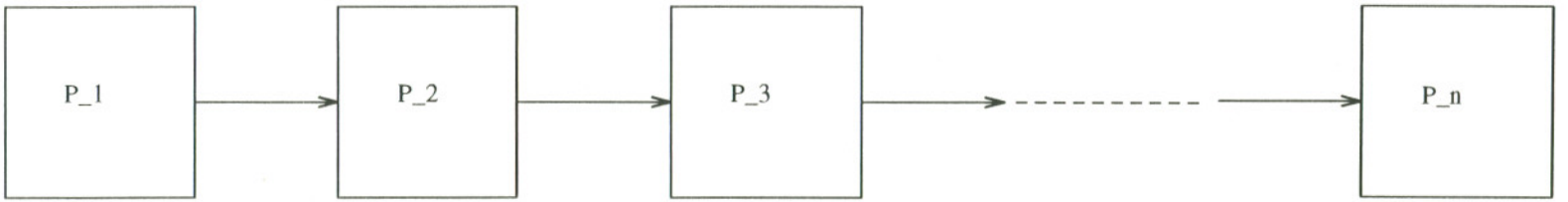


Fig. 1

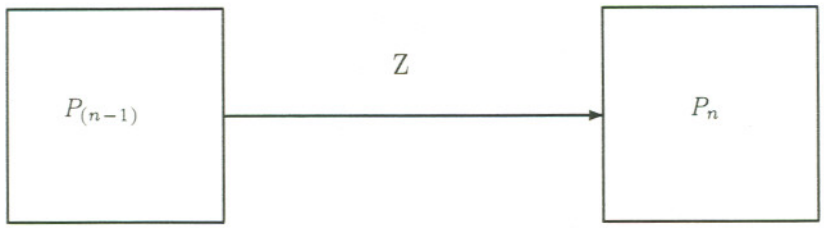


Fig. 2

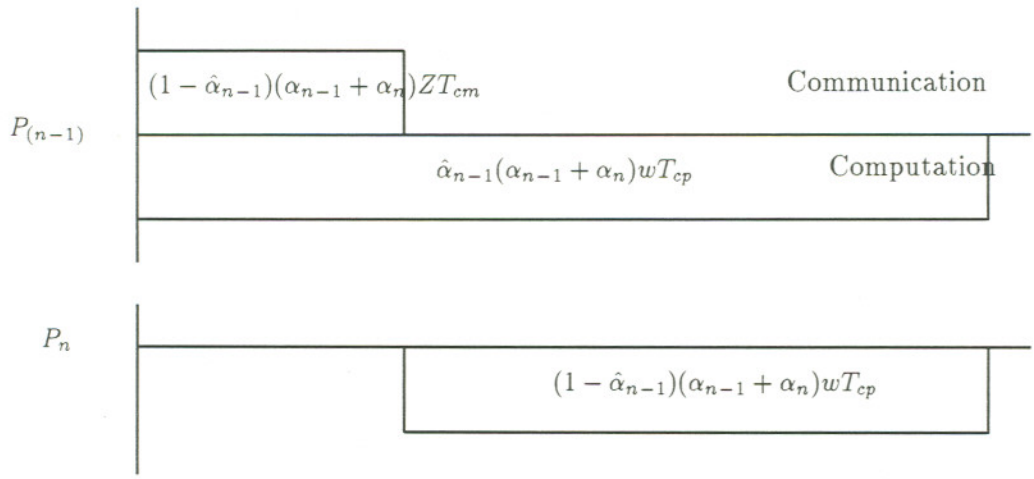
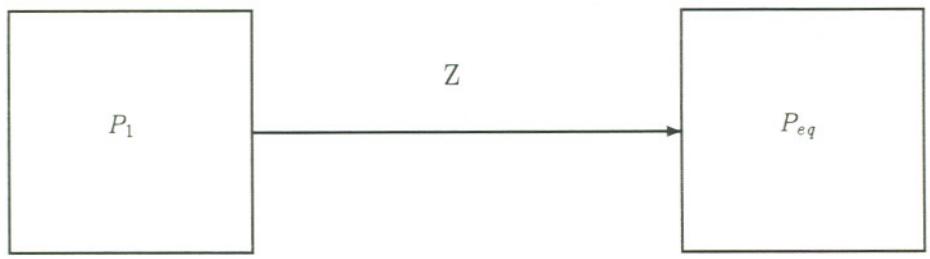


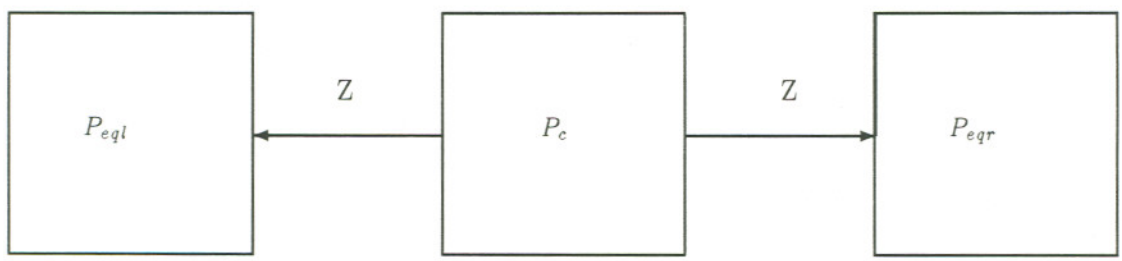
Fig. 3 The Timing Diagram



A Reduced Daisy Chain Where Load Originates

at Network Boundary

Fig. 4



A Reduced Daisy Chain Where Load Load Originates

at Network Interior

Fig. 5

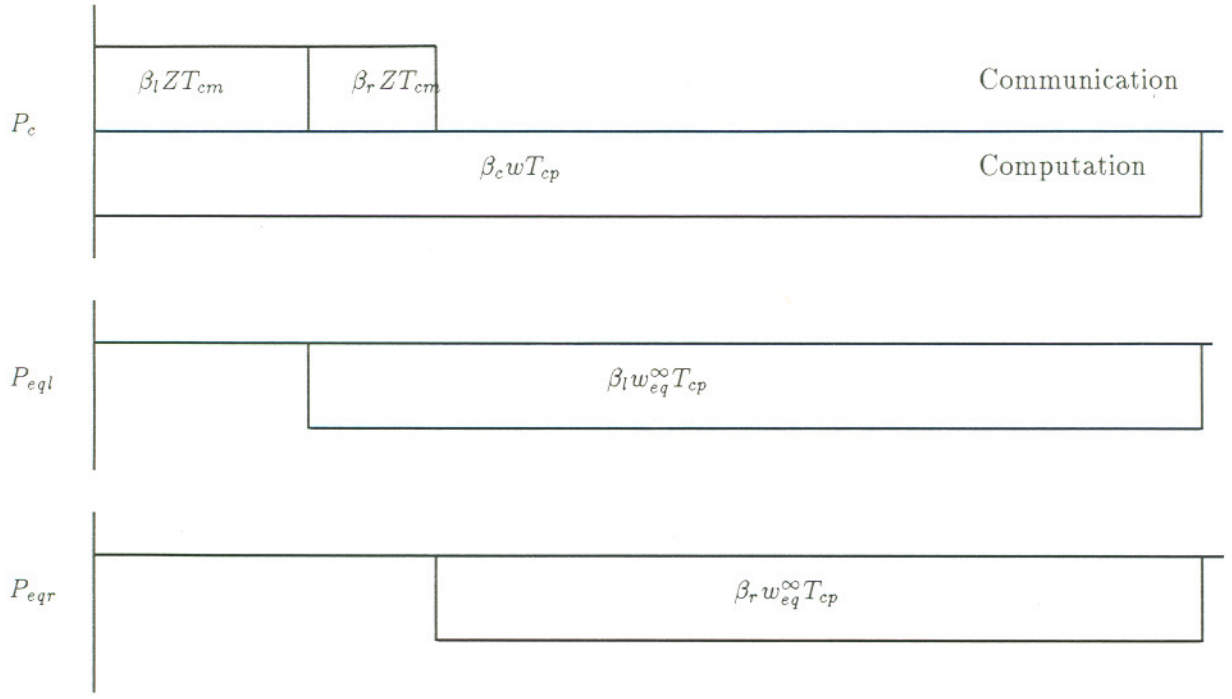

P_c
$\beta_l Z T_{cm}$
$\beta_r Z T_{cm}$
Communication
$\beta_c w T_{cp}$
Computation
P_{eql}
$\beta_l w_{eq}^{\infty} T_{cp}$
P_{eqr}
$\beta_r w_{eq}^{\infty} T_{cp}$

Fig. 6

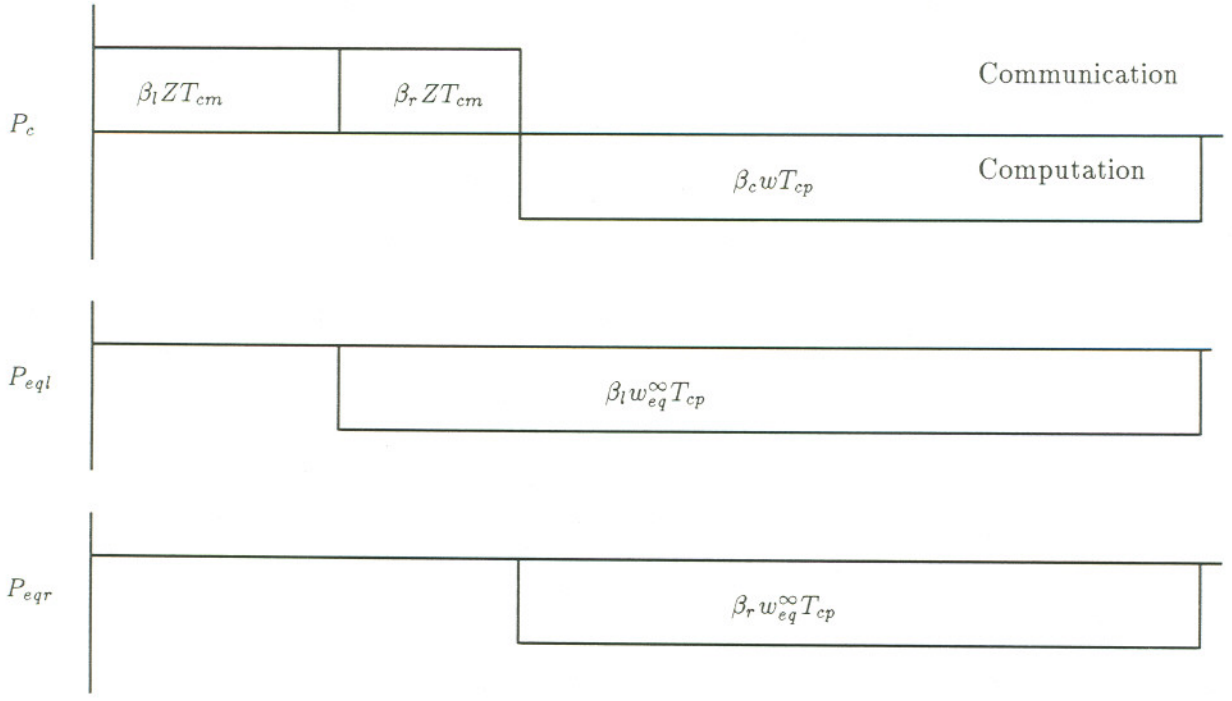
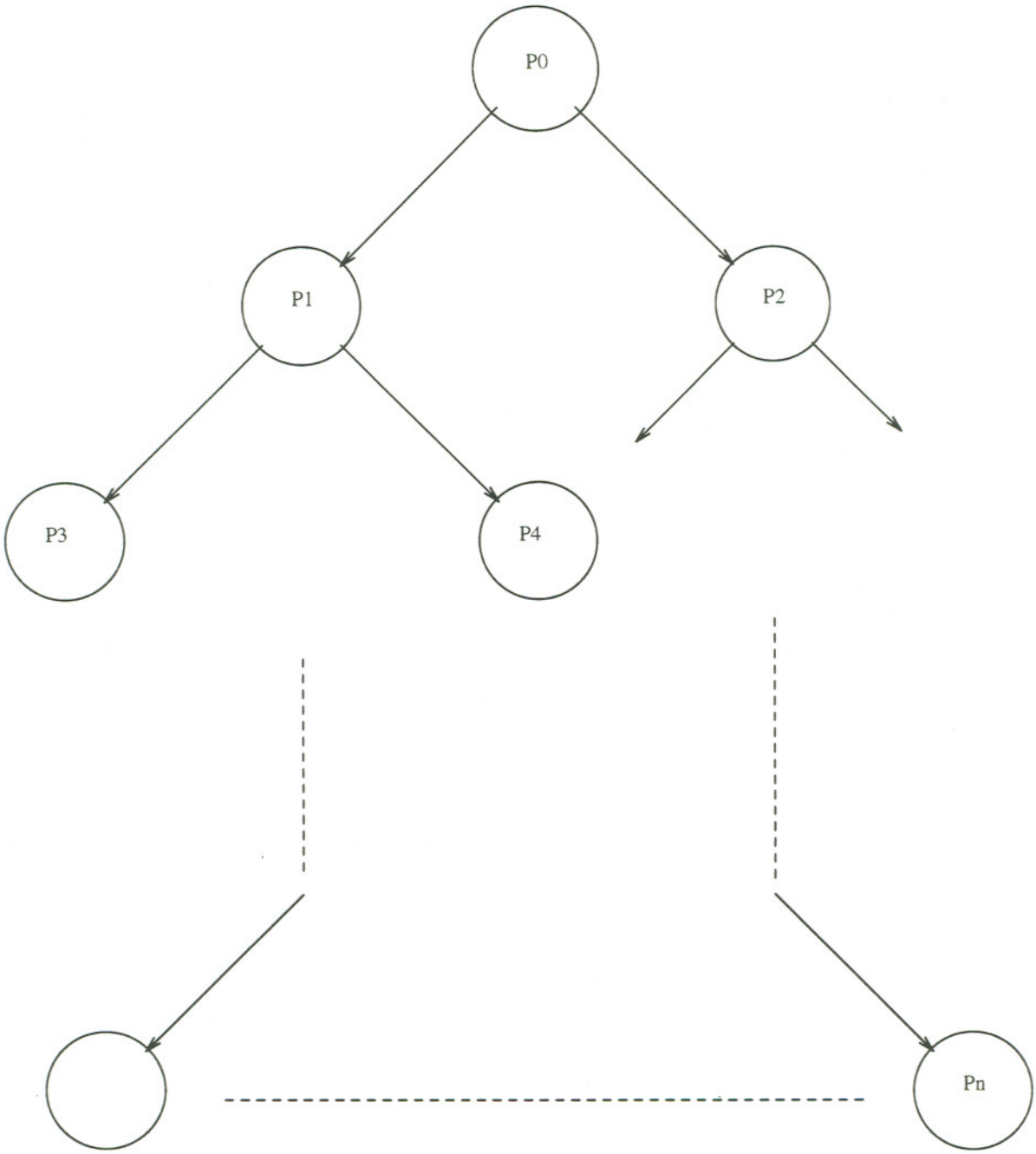

P_c
$\beta_l Z T_{cm}$
$\beta_r Z T_{cm}$
Communication
$\beta_c w T_{cp}$
Computation
P_{eql}
$\beta_l w_{eq}^{\infty} T_{cp}$
P_{eqr}
$\beta_r w_{eq}^{\infty} T_{cp}$

Fig. 7



Binary Tree Network

Fig. 8

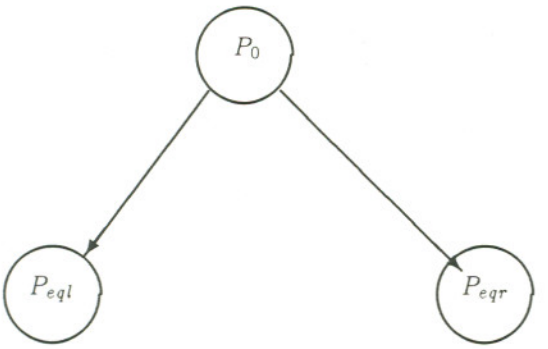


Fig. 9: Areduced Tree Network

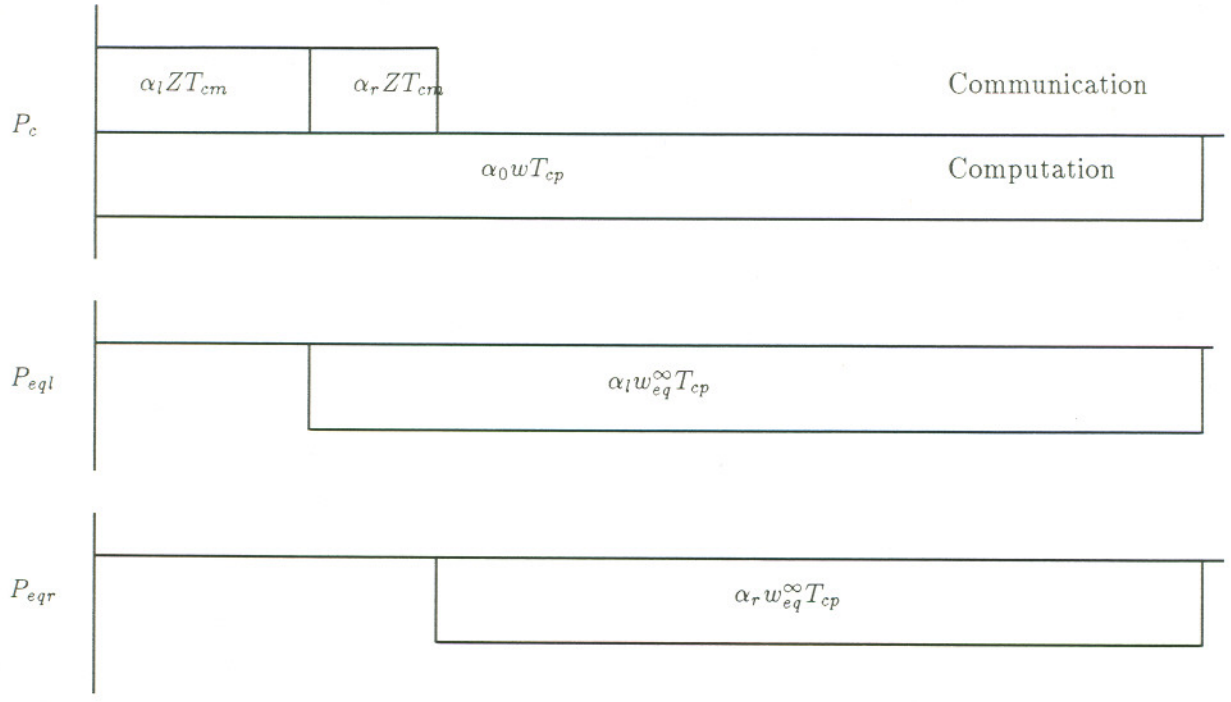

P_c
$\alpha_l ZT_{cm}$
$\alpha_r ZT_{cm}$
Communication
$\alpha_0 wT_{cp}$
Computation
P_{eql}
$\alpha_l w_{eq}^{\infty} T_{cp}$
P_{eqr}
$\alpha_r w_{eq}^{\infty} T_{cp}$

Fig. 10

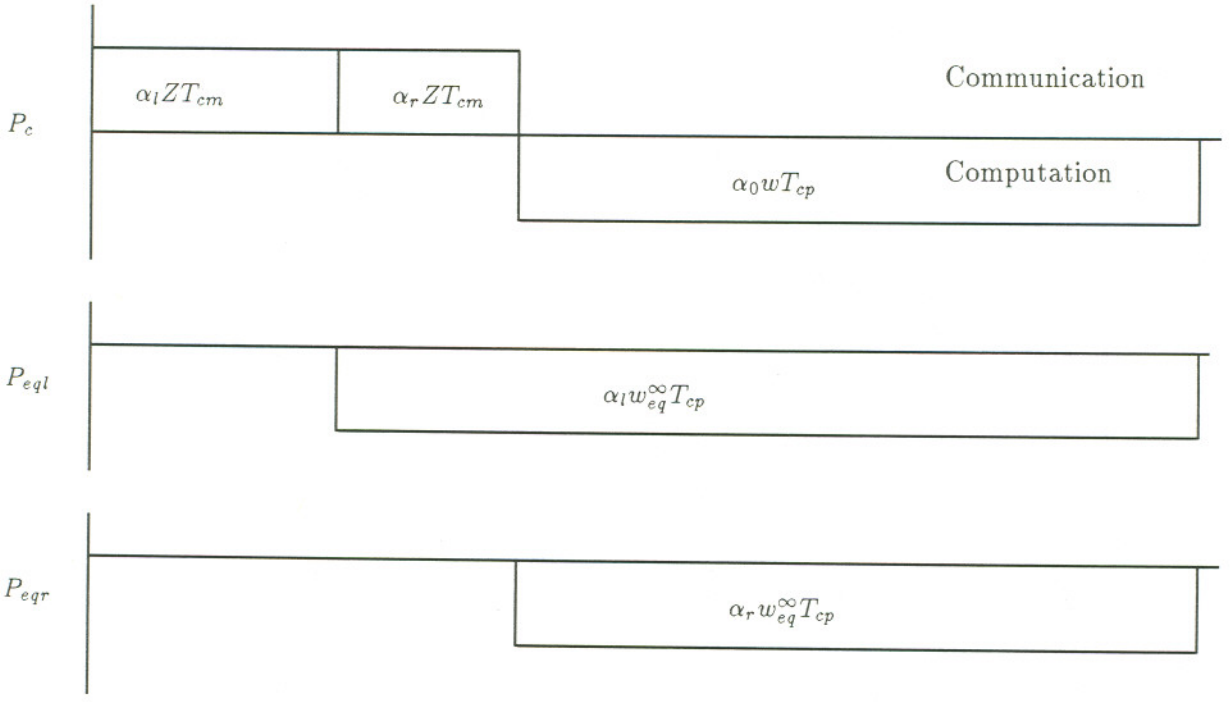

P_c
$\alpha_l ZT_{cm}$
$\alpha_r ZT_{cm}$
Communication
$\alpha_0 wT_{cp}$
Computation
P_{eql}
$\alpha_l w_{eq}^{\infty} T_{cp}$
P_{eqr}
$\alpha_r w_{eq}^{\infty} T_{cp}$

Fig. 11

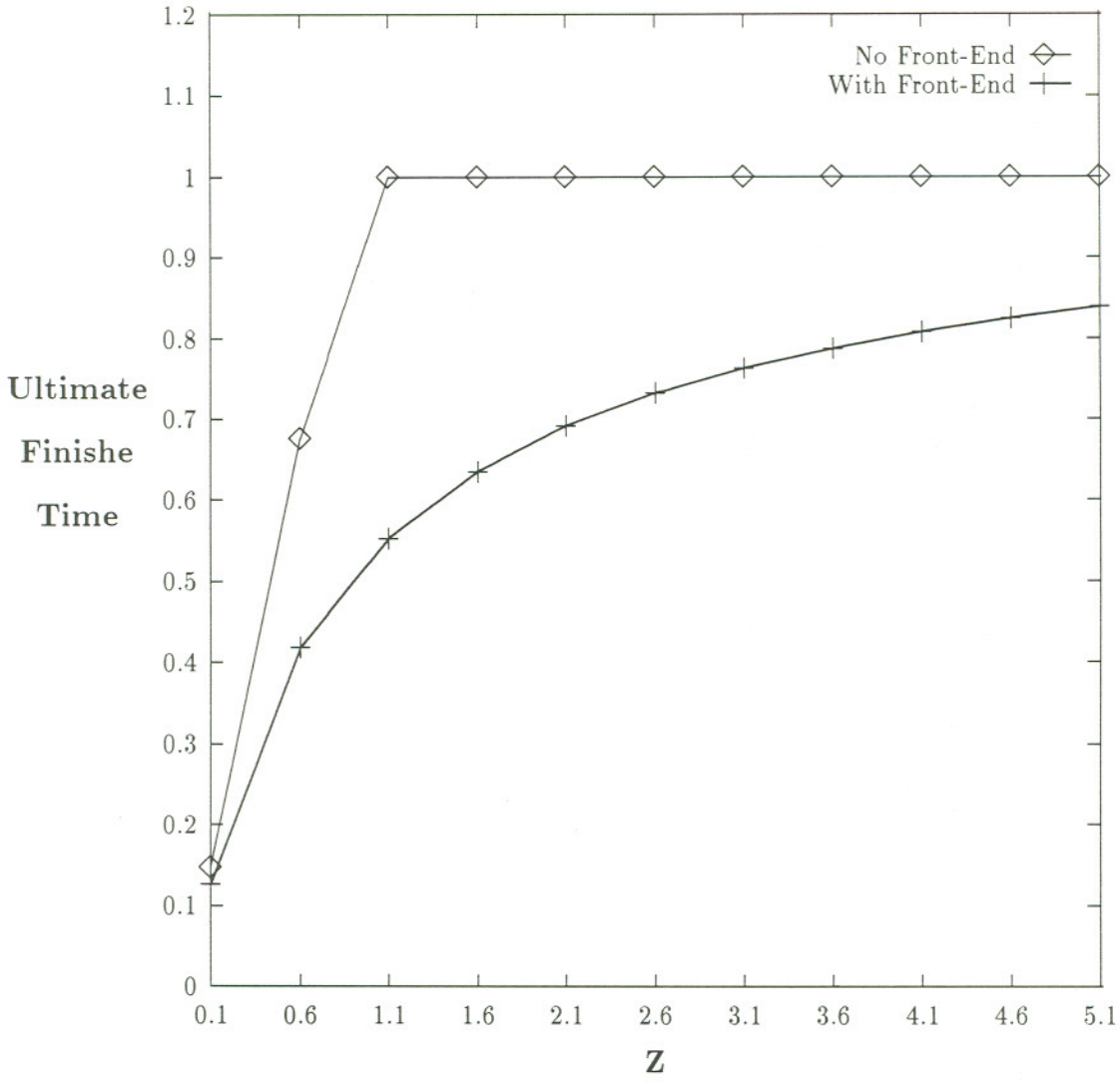


Fig. 12: Tree Network

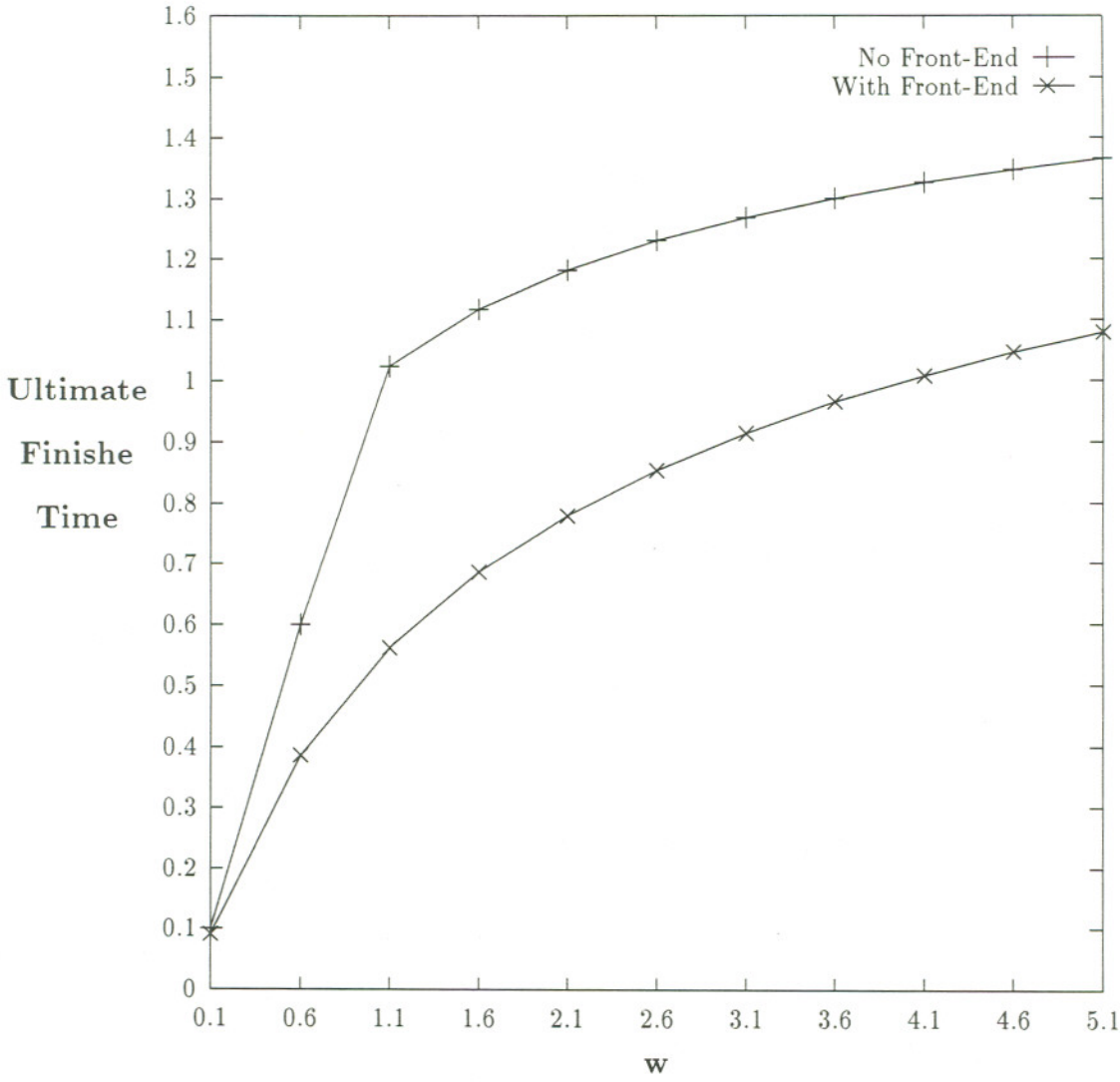


Fig. 13: Tree Network

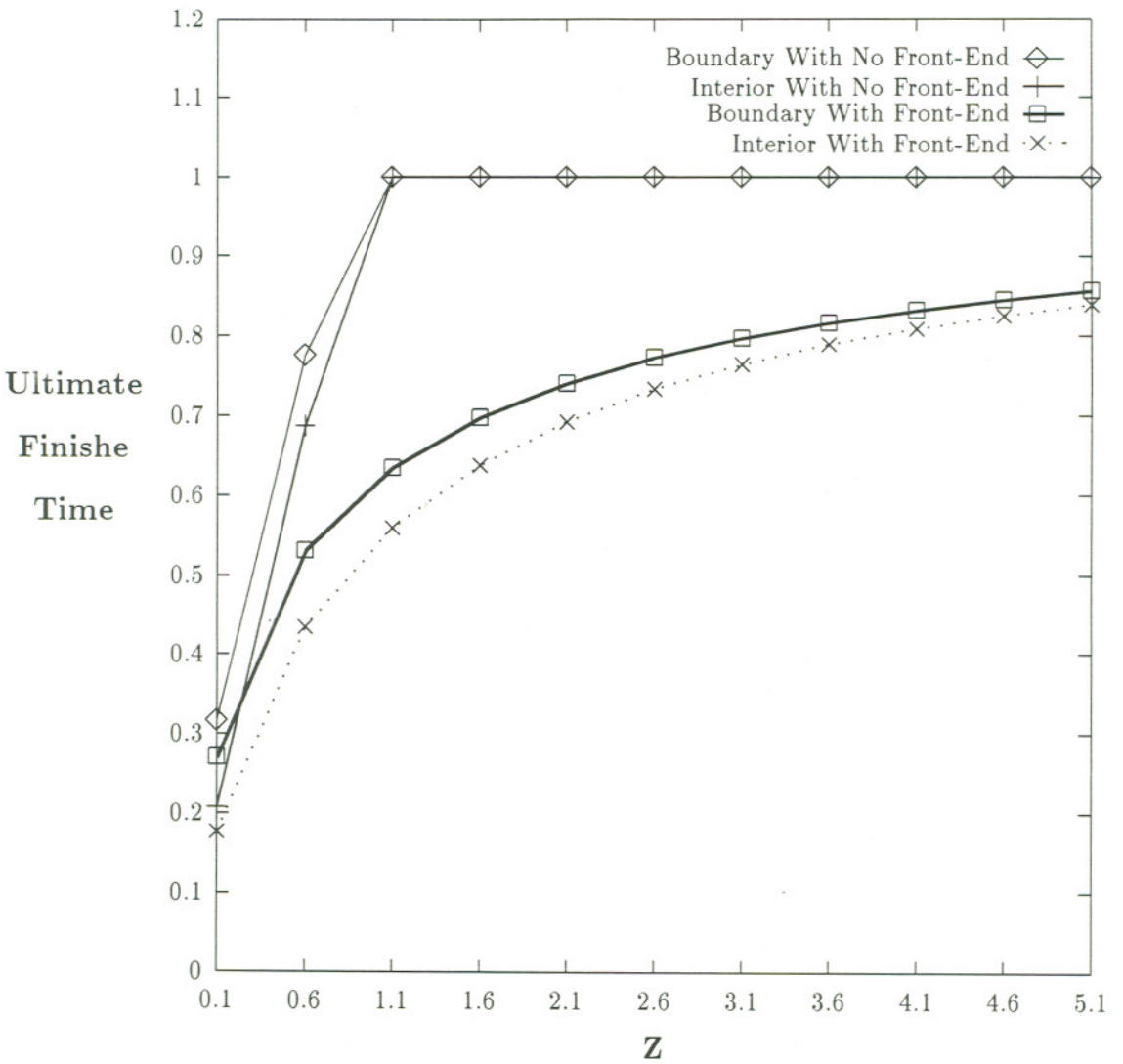


Fig. 14: Daisy Chain Network

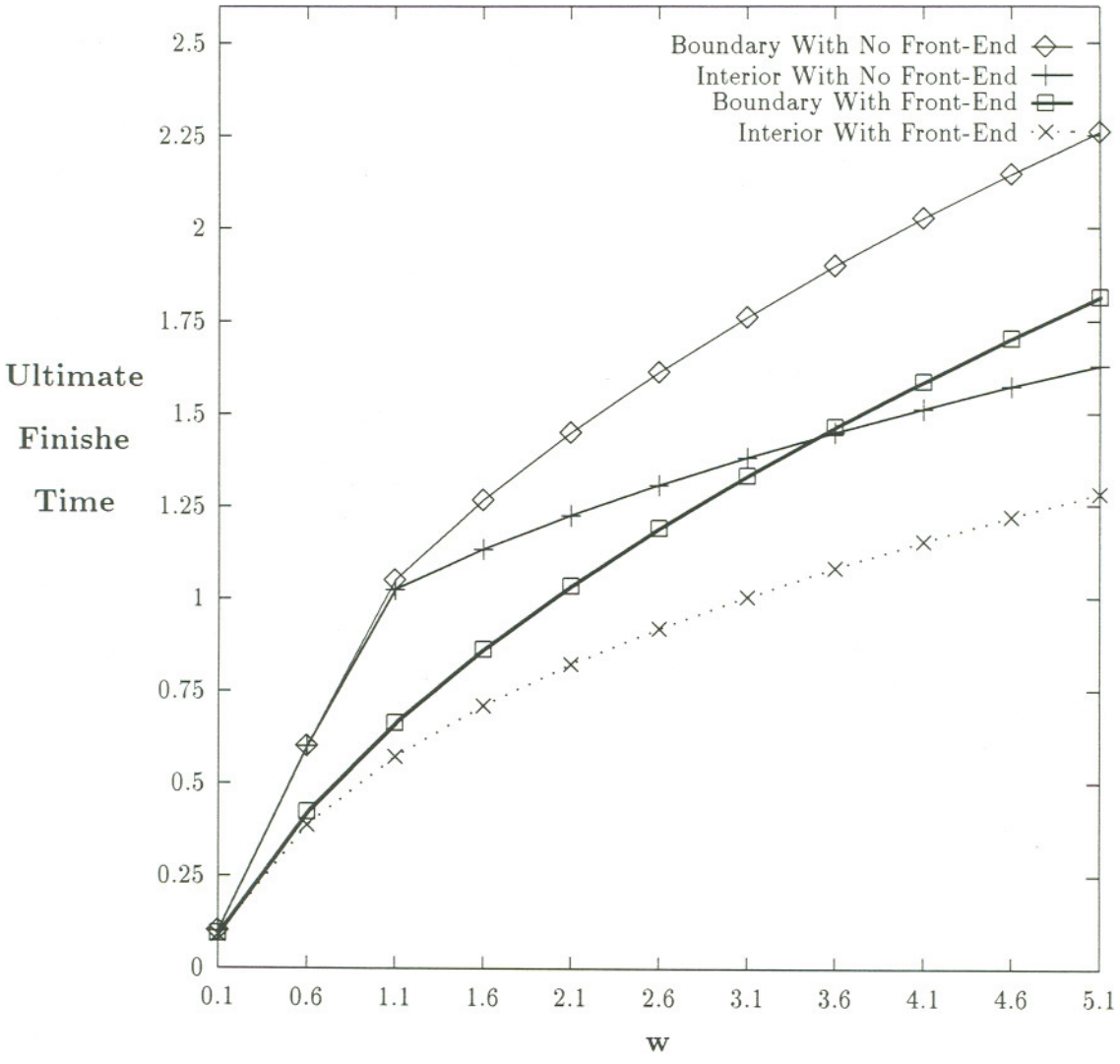


Fig. 15: Daisy Chain Network

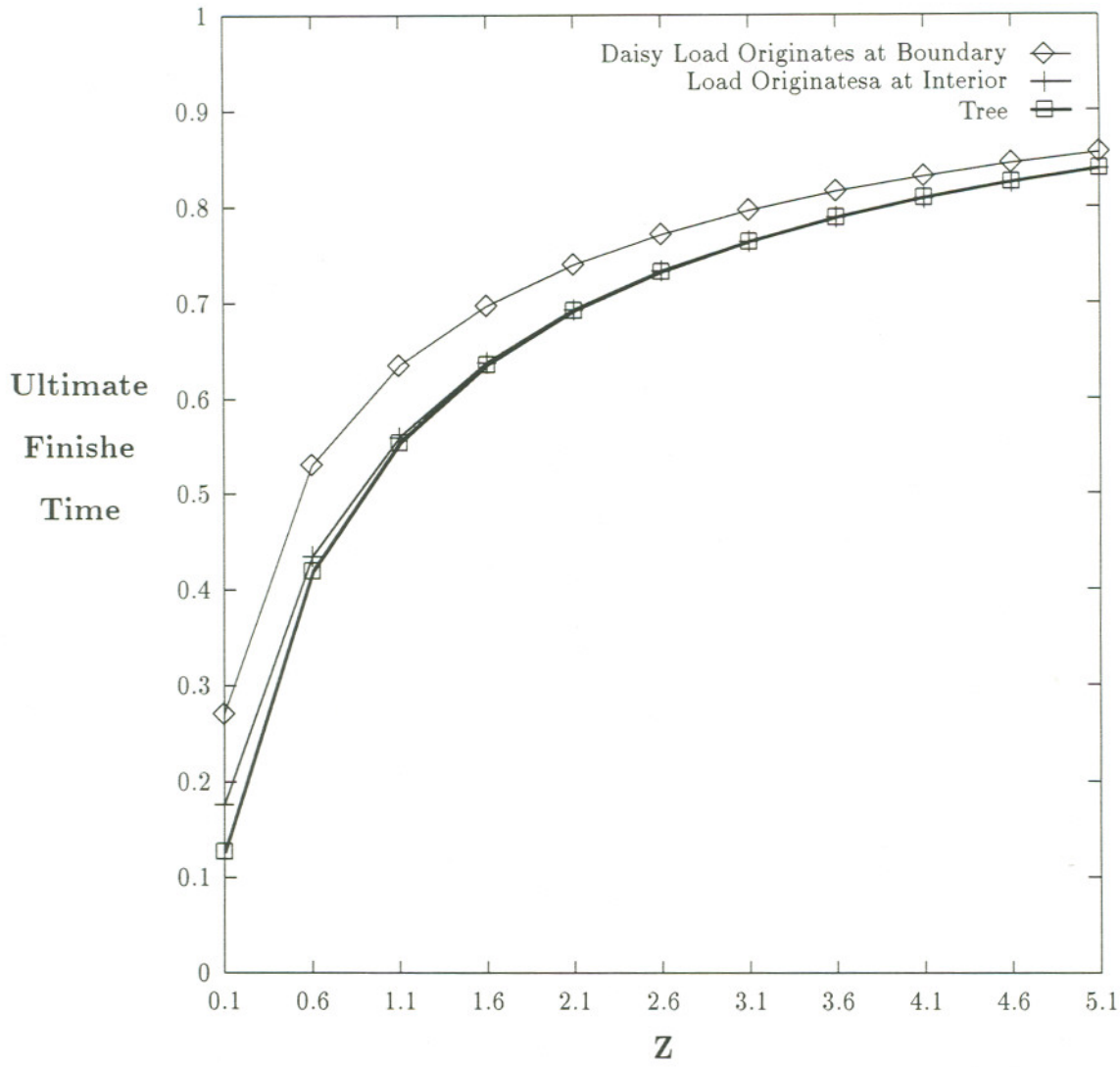


Fig. 16: Daisy Chain and Tree Networks with Front-End Processors

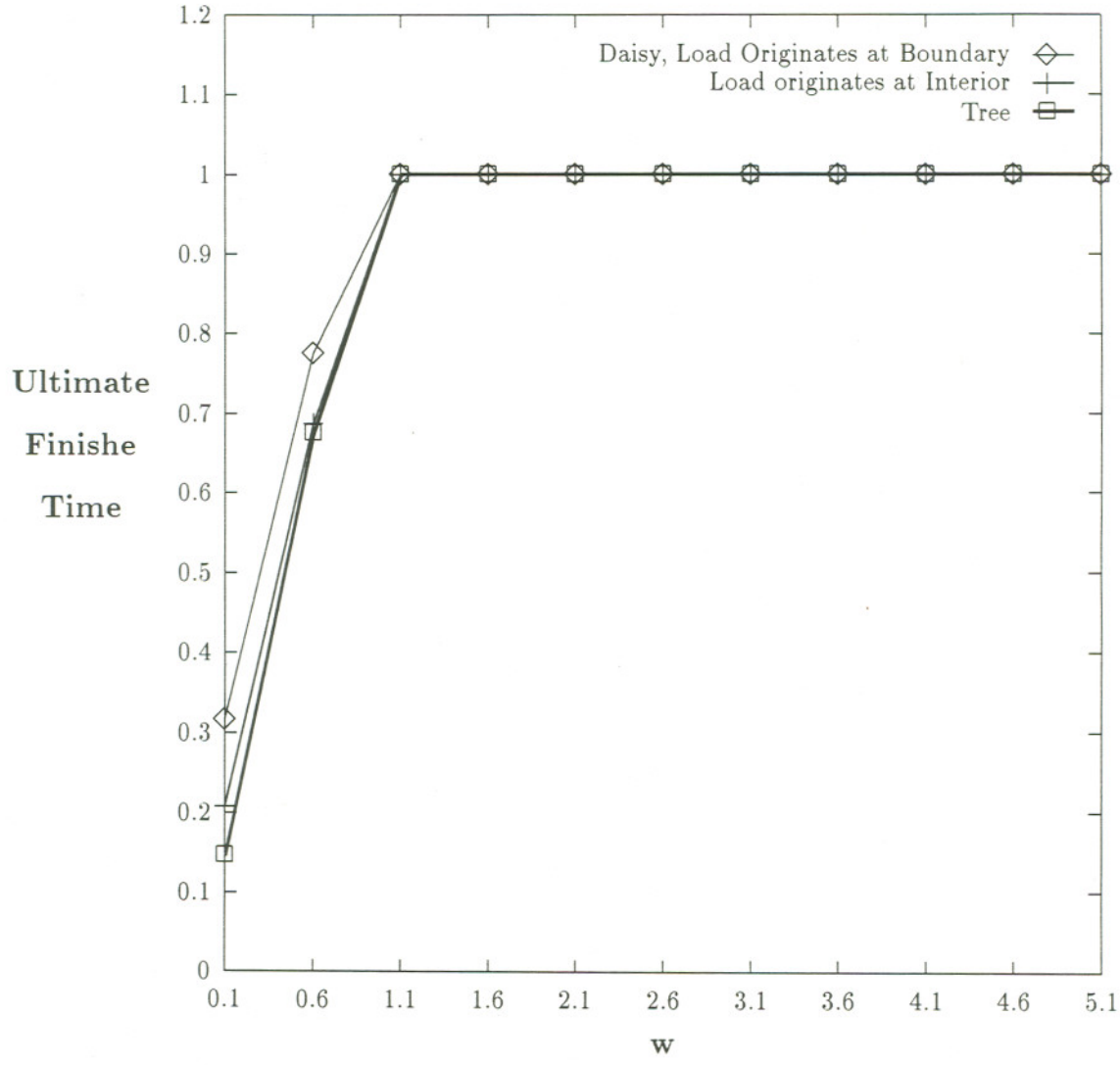


Fig. 17: Daisy Chain and Tree Networks with Front-End Processors

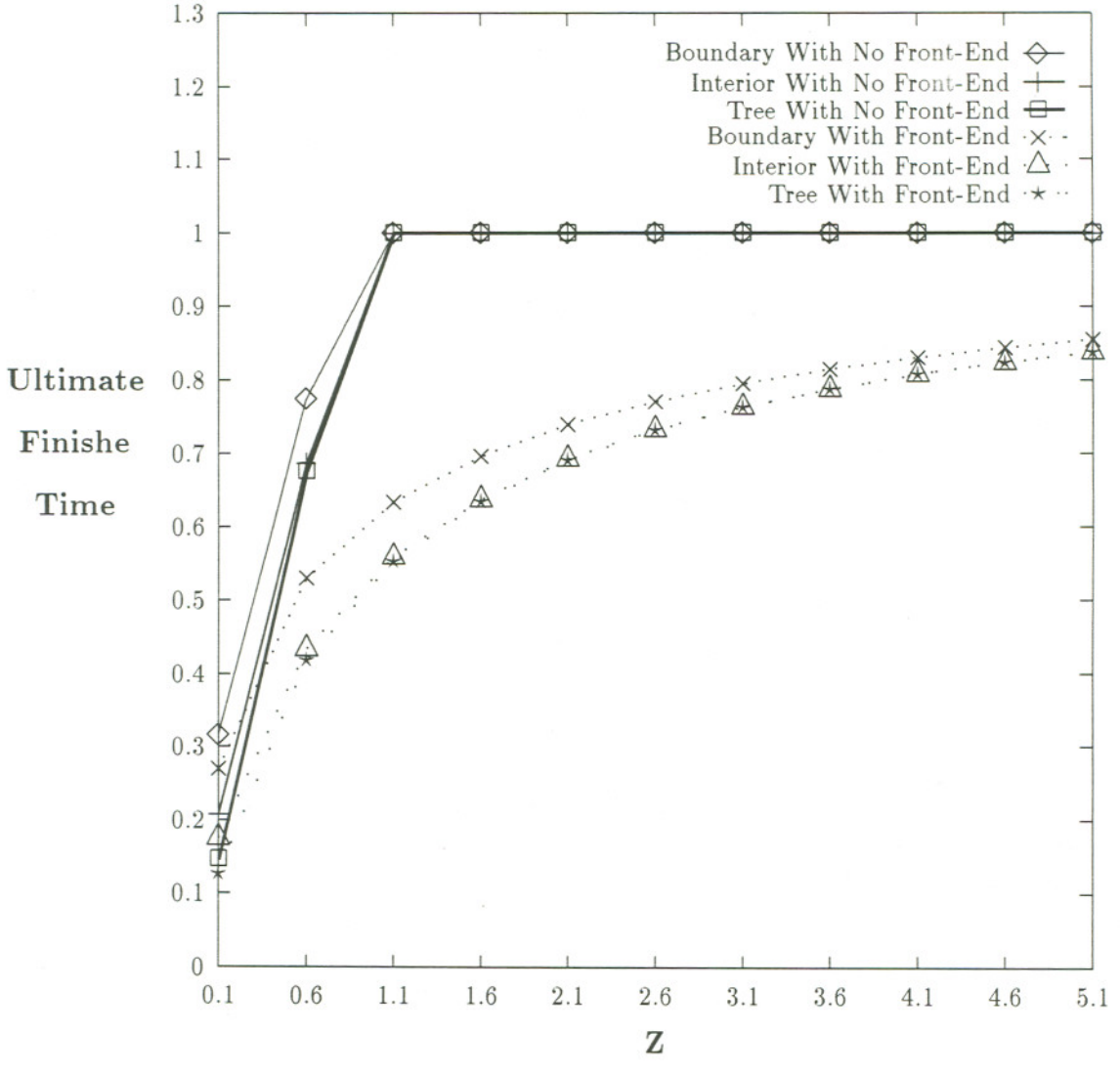


Fig. 18: Daisy Chain and Tree Networks