Theoretical Performance-Based Cost-Effectiveness of Multicomputers

R. G. BORN* AND J. R. KENEVAN†

* Department of Business Systems and Analysis, Northern Illinois University, DeKalb, IL 60115, USA
† Department of Computer Science, Illinois Institute of Technology, Chicago, IL 60616 USA

The CPU cycles that are stolen to relay messages can significantly affect the performance of a multicomputer system. This degradation in performance in turn affects the overall cost-effectiveness of such a system. This paper compares the cost-effectiveness of four multicomputer architectures that have received a great deal of recent attention: the bidirectional straight line, the unidirectional ring, the square mesh, and the binary hypercube. Cost-effectiveness is measured by finding the ratio of the total dollar cost of a multicomputer, based upon the total cost of processors and communication links, to the total potential utilization of the processors making up the system, where the potential utilization is the fraction of time a processor has to do useful processing after subtracting time spent handling messages. The smaller the value of this ratio, the greater the cost-effectiveness of the system.

Received April 1989, revised September 1991

1. INTRODUCTION

With the current widespread interest in parallel and distributed processing, the subject of multicomputers has received considerable attention in literature. This interest is quite natural when one considers the ever increasing demands that are being placed upon computer systems and the fact that there appears to be a fundamental limit to the computing power that can be compressed upon a single chip.

An early taxonomy that categorizes systems of interconnected computers based upon strategy of message transfer, transfer control, path structure, and specific system architecture, is given by Anderson and Jensen. In a survey by Feng, the space of interconnection networks is represented as the cross-product of sets of design decisions including operation mode, control strategy, switching methodology, and network topology.

The problem of optimizing the performance of multicomputer systems has been addressed by a number of researchers. Three common sense factors including the interconnection topology, the scheduling and mapping of the algorithm to the topology, and parallelism detection are pointed out by Agrawal and JanaKiram. Also discussed in Ref. 1 are several network characteristics (e.g. average distance, normalized average distance, and message density) that provide semi-quantitative means for comparing alternate topologies. A technique known as bottleneck analysis, discussed by Reed and Grunwald, allows determination of upper bounds on the rate at which a network is capable of routing messages from source to destination nodes. Performance characteristics (e.g. average message delay, message traffic density, total connection costs, number of connections per node, ease of message routing, and fault tolerance) of a number of static topologies are investigated by Wittie, with particular attention being paid to rate of increase of several key factors as the size of the network increases.

One area that has received little if any attention is that of comparing the theoretical cost-effectiveness of different multicomputer topologies. We attempt to break ground in this area by letting cost-effectiveness be a function of both a performance measure referred to as processor potential utilization (discussed in Refs 3, 4 and 5) and the total dollar cost of a multicomputer. Processor potential utilization, to be discussed in more detail in the following sections, is roughly the fraction of time a processor has to do useful processing after subtracting the time spent handling messages. Potential utilization is employed because of its usefulness in dealing with performance issues related to uniprocessor nodes, which will be considered in this paper. In addition, potential utilization will be seen to provide a very convenient framework in which to develop cost-effectiveness measures. The total dollar cost of a multicomputer will be based upon the cost of both the processors and the communication links.

We will be concerned with a class of multicomputers that may be described as having regular, static interconnection networks. By regular, it is meant that the interconnection scheme has identical neighbourhood relationships with the possible exception of boundaries, in agreement with the concept of regularity suggested in Ref. 2. By static, it is meant that the links that connect processors cannot be reconfigured to connect other processors. There are a wide variety of topologies that fit the regular static category. These include the one-dimensional linear array, the two-dimensional ring, star, tree, near-neighbour mesh, and systolic array, and the three-dimensional completely connected network, chordal ring, 3-cube, and 3-cube-connected cycle, all surveyed in Ref. 6. Three other interesting link-oriented topologies (i.e. the alpha network, hypercube, and the multitree structure) are discussed in Ref. 1.

Our discussion shall be limited to comparing the cost-effectiveness of four topologies that have received a great deal of recent attention: the bidirectional straight line (linear array), the unidirectional ring, the square mesh, and the binary hypercube, sometimes referred to as a binary n-cube. These are shown in Fig. 1. The straight line has been used in some pipeline architectures. The unidirectional ring has been widely used to connect multiple minicomputer and microcomputer systems in universities, laboratories, and industry settings. The
The cost-effectiveness of multicomputer topologies is discussed in the paper. Four types of multicomputer networks are selected for study: (a) A bidirectional straight line of \( m \) processors, (b) A unidirectional ring of \( m \) processors, (c) A square mesh of \( m \times m \) processors, and (d) 1-, 2-, and 3-dimensional binary hypercubes.

Square meshes have been used in some computers designed for the purpose of processing images (see Hillis).\(^7\) Meshes have also been used in solving second-order partial differential equations, and some very efficient sorting and matrix multiplication algorithms have been developed for these networks.\(^8\) One of the first designs using hypercube structure was the Cosmic Cube developed at the California Institute of Technology.\(^10\) Since that time, several companies including Intel Corporation, Floating Point Systems, and Thinking Machines Corporation, have developed computers based upon the hypercube structure. For example, the Connection Machine, manufactured by Thinking Machines Corporation, has a current maximum of 65,536 processors and has been used in a variety of applications in image processing,\(^7\) free-text database search,\(^11\) physical process simulation,\(^7\) and memory-based reasoning.\(^12\)

Toward our goal of comparing the cost-effectiveness of the four selected multicomputer topologies, section two will define the concept of processor potential utilization, will delineate the assumptions behind our theoretical models, and will provide an outline of the method for deriving processor potential utilization (the details of which can be found in Refs 3, 4 and 5). Section three shows, through a simulation perturbation analysis, that one of the critical assumptions can be significantly relaxed and yet cause little change in the results. Finally, in section four, we compare the cost-effectiveness of the four topologies.

2. Definitions, Assumptions and Analytic Method

This section of the paper begins with a definition of the concept of processor potential utilization. Next, the assumptions behind our theoretical model are discussed, and then the derivation of processor potential utilization is summarized for two separate cases—constant rate message generation and message generation rate proportional to potential utilization. In each of the two cases, the basic results for the bidirectional straight line, unidirectional ring, square mesh, and binary hypercube will be summarized.

2.1 Definition of processor potential utilization

An important concept that will be used to determine the theoretical performance-based cost-effectiveness of the topologies under consideration is that of processor potential utilization. If \( f_t \) is the fraction of time that...
processor :i: spends processing messages, then :f_i: can be partitioned into three components as follows:

1. \( f_i \), the fraction of time that processor :i: spends processing messages it must send because it is the originating processor,
2. \( f_i \), the fraction of time that processor :i: must spend relaying messages because it is an intervening processor between the originating and destination processor, and
3. \( f_i \), the fraction of time that processor :i: must spend accepting messages for which it is the destination processor.

We then define processor potential utilization, denoted by \( U_i \), as the fraction of time remaining for processor :i: to do useful work, after subtracting the time spent handling messages:

\[
U_i = 1 - f_i = 1 - (f_i + f_i + f_i).
\]  

It is desirable to compute the steady-state values of \( U_i \) for \( 1 \leq i \leq m \), where \( m \) is the number of processors in the network.

### 2.2 Model assumptions

The assumptions that follow form the basis for the theoretical performance-based cost-effectiveness results to be discussed in section four:

1. **Uniform message routing.** For each originating processor for a message, a destination processor is randomly selected uniformly from the set of processors excluding the originating processor. Because this assumption does not take spatial locality into account by suggesting any kind of an intelligent scheduling mechanism whereby a processor sends messages only to processors that are nearby, it is more likely to provide an upper limit for relaying message traffic. It is also possible that in a large scale network, a hashing function might be used to locate tasks on processors, thus justifying this assumption of uniformly distributed destination nodes.

2. **Infinite bandwidth.** The channels connecting the processors in the network have infinite bandwidth, and the speed of the signals in the channels is infinitely fast. In other words, it is assumed that the channels do not provide bottlenecks for the relaying of messages. Adapting the theoretical model of this paper to account for finite bandwidth would be extremely difficult. However, adapting the simulation model to account for finite bandwidth would be quite straightforward, particularly in view of the excellent communication network simulation software (e.g. COMNET II.5) that is now available commercially. The authors are currently investigating the effects of relaxing this assumption in simulation analyses.

3. **Shortest path routing.** Messages are always routed over a shortest path connecting the source and destination processors.

4. **Independence of message generation processes.** The message generation process at any node is independent of the message generation process on any other node.

5. **Unbounded buffering capacity.** Each node has unbounded buffering capacity for holding messages.

6. **Packet switching.** Message transmission is accomplished through packet switching.

7. **Uniprocessor nodes.** Each node contains one processor responsible for both communication and processing. While many real networks contain an additional communication processor, this assumption will tend to provide a worst-case analysis of the performance of multicomputer systems.

### 2.3 Case 1 - Constant message generation rate

In this case it is assumed that messages are generated at each processor in such a way that the time between successive messages is an exponential random variable, and that the mean message generation rate is given by \( R_c \) (standing for Rate under the assumption of constant message generation rate). We assume that a processor requires a time \( t_i \) to process the sending of a message as an origination processor, a time \( t_i \) to process the relaying of a message as an intervening processor, and a time \( t_i \) to process the final acceptance of a message as a destination processor.

In the steady-state, any one of the processors would average sending one message to each of the remaining \( m - 1 \) processors in a time given by \((m - 1)/R_c\), since \( 1/R_c \) would be the intermessage time. During the time \((m - 1)/R_c\), processor \( i \) originates \( m - 1 \) messages each requiring time \( t_i \) for \( i \) to send, relays \( M_i \) messages each requiring time \( t_i \) for \( i \) to relay, and receives \( m - 1 \) messages as a destination processor, each requiring time \( t_i \) for processor \( i \) to accept. Combining these facts with the definition of processor potential utilization given in equation (1), it is found that

\[
U_i = 1 - R_c[(m - 1) t_i + M_i t_i + (m - 1) t_i]/(m - 1).
\]  

The value of \( M_i \) is a function of both the architecture and the routing algorithm used to relay messages from the originating to the destination processor. Some tedious algebraic derivations, detailed in Refs 3 and 4, give the following formulas for \( M_i \):

\[
M_i = 2(m - 1)(i - 1), \quad \text{straight line of } m \text{ processors};
\]

\[
M_i = (m - 2)(m - 1)/2, \quad \text{unidirectional ring of } m \text{ processors};
\]

\[
M_i = (n - 2)2^{n-1} + 1, \quad \text{binary hypercube of } m = 2^n \text{ processors};
\]

\[
M_i = 1 - 2n + 2cn + 2rn - 2c^2n + 2cn^2 - 2r^2n + 2rn^2 - 3n^2, \quad \text{square mesh of } m = n^2 \text{ processors}.
\]

Regarding equation (6) for the square mesh, it should be noted that \( M_{i,c} \) is the number of messages relayed by the processor in row \( r \) and column \( c \) during a time when each processor in the network generates exactly \( m - 1 \) messages. It should also be noted that the calculation of \( M_{i,c} \) is based upon what the authors refer to as perimeter routing, in which a message is always routed deterministically along the perimeter of the rectangle containing the source and destination processor.
2.4 Case 2 – Message rate proportional to processor potential utilization

In this case it is assumed that messages are generated at each processor in such a way that the rate at which any processor generates messages is proportional to the processor's current potential utilization. The proportionality constant, denoted by \( K \), represents the rate at which a processor would generate messages if its current potential utilization were 1. The value of \( K \) is assumed to be the same for every processor in the network. Thus, if two distinct processors have current potential utilizations of 0.6, then both would generate messages at the rate 0.6\( R \). This assumption carries with it the notion that a processor is more likely to generate messages if it is not busy relaying messages for other processors. We associate the same three processing time costs (\( t_s, t_t \), and \( t_c \)) with message handling that were used in Case 1.

Since processor \( i \) must process the sending of all messages that it generates, then at steady-state for \( 1 \leq i \leq m \),

\[
f'_i = U_i R t_i.
\]  

(7) 

Processor \( i \) is the destination processor for \( 1/(m-1) \) of the messages generated by each of the remaining \( m-1 \) processors. Therefore, for \( 1 \leq i \leq m \),

\[
f'_i = \sum_{k=1}^{m} U_k R t_i \left( 1/(m-1) \right).
\]  

(8) 

The equation for \( f'_i \) is a bit more complicated as it involves the routing algorithm used. The general form of the equation for \( f'_i \), where \( 1 \leq i \leq m \), at steady-state is

\[
f'_i = \sum_{k=1}^{m} U_k R t_i \ p_i.
\]  

(9) 

where \( p_i \) is the probability that processor \( i \) relays a message originating at processor \( k \).

Consideration of equations (1) and (7) to (9) suggests that the solution to the problem amounts to solving a system of \( m \) equations for the \( m \) unknowns \( U_1, U_2, U_3, \ldots, U_m \). The unidirectional ring and binary hypercube, being symmetric networks (since each of the processors sees the same view of the network), provide, after some tedious algebra, closed form expressions for potential utilization:

\[
U_i = \frac{1 + R t_i + R t_i (m-2)/2 + R t_i}{1}, \quad \text{unidirectional ring};
\]  

(10) 

\[
U_i = \frac{1 + R t_i + R t_i [(n-2)2^{n-1} + 1]/(2^n-1) + R t_i}{1}, \quad \text{binary } n\text{-cube}.
\]  

(11) 

The solution for the asymmetric bidirectional straight line and the square mesh can be expressed in matrix form, as detailed in Refs 3 and 5; the results are summarized in Appendix A and Appendix B.

3. PERTURBATION ANALYSIS

The major assumption in section 2.4 was that message generation rate be proportional to a processor's current potential utilization. Just how critical is this assumption? Would simulation results without any perturbation from this assumption agree with simulation results under strong perturbations from this assumption? These are the kinds of questions that we address in this section.

Each processor \( i \) will be assumed to generate messages at a rate equal to \( c_i R U_i \), where \( c_i \) is uniformly distributed on the interval \( 1 - x \leq c_i \leq 1 + x \), where \( 0 < x < 1 \). Here, \( R \) is the rate of message generation for any processor at 100% potential utilization and no perturbation. \( x \) will be called the perturbation factor. For example, if \( x = 0.25 \), then the values of \( c_i \) will be distributed uniformly on the interval [0.75, 1.25]; each processor would deviate by as much as 25% from its generation rate under strict adherence to the assumption that rate be proportional to potential utilization.

To measure the effect of the perturbations, a quantity which we shall call \( \text{AbsDevPert} \) is defined for each processor \( i \). It is the absolute value of the difference

<table>
<thead>
<tr>
<th>Table 1. Perturbation statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perturbation factor, ( x )</td>
</tr>
<tr>
<td>Topology</td>
</tr>
<tr>
<td>---------</td>
</tr>
<tr>
<td>Straight line</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Ring</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Binary hypercube</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Square mesh</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
between the potential utilization obtained with perturbation and the potential utilization obtained without perturbation in simulation runs:

\[ \text{AbsDevPert}_i = \left| \text{Stem w/pert} U_i - \text{Stem w/o pert} U_i \right|. \tag{12} \]

In equation 12, \( \text{Stem w/pert} U_i \) is the potential utilization of processor \( i \) under a simulation run with perturbation, whereas \( \text{Stem w/o pert} U_i \) is the potential utilization of processor \( i \) under a simulation run without perturbation (i.e., when the perturbation factor \( x = 0 \)). It is further assumed that the potential utilizations are expressed as percents in equation 12 as opposed to a number between 0 and 1.

In order to compare the effects of different values of the perturbation factor on a network as a whole, it is useful to calculate both the mean and the standard deviation of the \( \text{AbsDevPert}_i \) values of all the processors in a network. These shall be referred to as the Mean AbsDevPert and StdDev AbsDevPert, respectively.

Discrete-event simulation runs were made for each of the four topologies using the following values of the perturbation factor \( x = 0.10, 0.25, 0.50, 0.75 \) and 0.90. The simulations were written using GPSS (General-Purpose Systems Simulator) for 64-processor networks, with \( t_s = t_{a} = 16 \) time units and \( t_r = 32 \) time units. These runs were made for two separate values of \( R: 0.0001 \) per time unit and 0.001 per time unit. Thus, if the time unit is a millisecond, \( R = 0.0001 \) would correspond to one message every 10 seconds, and \( R = 0.001 \) would correspond to a message generated every second, at 100% potential utilization. While these rates are low compared to what might be generated in fine grain parallelism, higher rates than this are not beyond the scope of the type of networks modelled in this paper. The low rates were selected simply because higher rates use up significantly more CPU time in the GPSS simulations, particularly for networks with 64 or more processors.

Table 1 contains Mean AbsDevPert and StdDev AbsDevPert statistics for the perturbation runs, providing measures of central tendency and variability. Figure 2(a) and 2(b) provide a more visual interpretation of the statistics by showing graphs of Mean AbsDevPert versus the perturbation factor \( x \) for \( R = 0.0001 \) and 0.001 per time unit, respectively. Figure 2 shows that for the lower \( R \) value of 0.0001 per time unit, the Mean AbsDevPert values are less than three-tenths of one percent, even with perturbation factors as high as 0.90! With the higher value of \( R \), the Mean AbsDevPert values are less than 1.6% for perturbation factors as high as 0.9! Although this result may seem surprising at first, it seems reasonable if one remembers that the \( c_i \) values are uniformly distributed on the interval \([1-x, 1+x]\). It is possible that a processor at some point along the straight line, for example, may have a \( c_i \) value close to \( 1-x \), whereas a processor next to it on the line may have a \( c_i \) value close to \( 1+x \). Therefore, strong perturbations from the assumption that message rate be proportional to potential utilization have little effect, suggesting that real networks not adhering strictly to this assumption may still behave in reasonable accordance to our theoretical model.

Figure 2 also reveals that the binary hypercube seems to be the least sensitive of the four topologies to perturbations. In addition, the data of Table 1 reveals that the variability in the AbsDevPert, values, as measured by StdDev AbsDevPert, is lowest for the binary hypercube. This analysis shows that the binary hypercube exhibits some very nice performance characteristics in addition to those normally reported for this topology.

4. COMPARATIVE COST-EFFECTIVENESS OF THE NETWORKS

In this section we consider first the total cost of a network, then the cost per unit total potential utilization as a means for comparing cost-effectiveness of different topologies. Finally, two examples shall be presented to illustrate the procedure. The first example will consider the case of constant message generation rate, while the second example will consider the case where message generation rate is proportional to a processor's current potential utilization.

4.1 Total cost of a network

The total cost of a given network can be approximated by considering a reasonable subset of network attributes.
Suppose that this subset contains the following four attributes:

- \( N_p \), the number of processors (or nodes),
- \( N_L \), the number of communication links,
- \( C_p \), the cost of a processor (or node),
- \( C_L \), the cost of a communication link.

It is assumed that the cost of ports is included in the cost of the link itself. The cost of a processor and the cost of a communication link can be measured in any convenient monetary unit. The total cost of a network, \( \text{TotCost} \), is then defined by the following:

\[
\text{TotCost} = N_p C_p + N_L C_L. \quad (13)
\]

Table 2 contains formulas for \( N_p \) and \( N_L \) for each of the topologies considered in this paper. The square mesh and binary hypercube deserve special comment. For the square mesh, the number of horizontal or vertical links is \( n(n-1) \) as can readily be seen by studying Fig. 1(c). For the binary hypercube, a study of Fig. 1(d) reveals the recursive formula for the number of links \( L(n) \) in a binary \( m \)-cube is given by \( L(n) = 2L(n-1) + 2^{m-1} \). This recursive formula can be used in conjunction with mathematical induction to quickly verify the formula given in Table 2 for the number of links in a binary hypercube.

### 4.2 Cost per unit total potential utilization

A more useful measure than total cost is needed for comparing network topologies. This is particularly true when considering networks for which overhead due to message-passing is a degrading factor. The measure that we suggest here is the cost per unit total potential utilization. This measure shall be denoted by \( \text{CperTPU} \) (Cost per Unit Total Potential Utilization) and is given by the formula:

\[
\text{CperTPU} = \frac{\text{TotCost}}{\sum_{i=1}^{m} U_i} \quad (14)
\]

Alternatively, if \( U_{avg} \) is the average potential utilization of the processors in a network, then \( \text{CperTPU} \) may be expressed by the formula:

\[
\text{CperTPU} = \frac{\text{TotCost}}{m U_{avg}} \quad (15)
\]

An advantage of defining cost-effectiveness in terms of the cost per unit total potential utilization rather than the cost per unit average potential utilization is that it would allow one to compare networks of different size. Nevertheless, average potential utilization is a convenient way to express the potential utilization of an average processor in a network.

Table 3 contains formulas for the average potential utilization of the processors in the four networks under consideration based upon the constant message generation rate assumption. The formulas are also based upon the simplifying assumption that the processing time costs \( t_p^R \), \( t_c \), and \( t_s \) are dependent, with \( t_s = t_p^R / 2 \), \( t_c = t_p^R \), and \( t_s = t_p^R / 2 \). Thus, a relaying time cost \( t_p^R \) is assumed that is twice as great as the time cost associated with the initial sending of a message by the originating processor and is twice as great as the time cost associated with the acceptance of the message by the final destination processor. The formulas for the ring and hypercube are easily derived by combining equation 2 with equations 4 and 5, respectively, and remembering that these two topologies are symmetric. The formulas for the asymmetric cases of the straight line and square mesh are found by summing \( U_i \) over all processors and then dividing by the number of processors in the network. The algebra is tedious but is simplified by using some well-known summation formulas.

### 4.3 Crossover points related to cost-effectiveness

It is of practical importance to determine when the cost-effectiveness of a given topology overtakes the cost-effectiveness of an alternate topology, subject to a given set of parameters that are held constant and another parameter that is allowed to vary. The topology that is more cost-effective for a given value of the variable parameter must have the smaller \( \text{CperTPU} \) value. If it is desired to compare topology \( x \) to topology \( y \), then the crossover point(s) can be found by setting the \( \text{CperTPU}_x \) of topology \( x \) equal to the \( \text{CperTPU}_y \) of topology \( y \):

\[
\text{CperTPU}_x = \text{CperTPU}_y \quad (16)
\]

and solving the resultant equation for the variable parameter. Two examples shall now be considered to illustrate the technique involved.

Example 1. Suppose that it is desired to determine the most cost-effective network among the four being studied as \( t_p^R \) varies, under the constant message generation rate assumption. It is further assumed that the message generation rate is the same for all of the topologies and
R. G. BORN AND J. R. KENEVAN

Figure 3. CperTPU versus \( t_pR_c \) for constant message generation rate assumption.

Figure 4. CperTPU versus \( R \) for message rate proportional to potential utilization.

each of the networks has 64 processors. Finally, it is assumed that the cost of a processor, \( C_p \), and the cost of a link, \( C_L \), are both one arbitrary monetary unit.

Figure 3 shows a graph of CperTPU versus \( t_pR_c \) for each of the four topologies under the conditions stated for this example. The six crossover points have been labelled with letters 'a' to 'f'. The crossover points a, c and f are of particular interest in this example. In Fig. 3, \( v_a, v_c \) and \( v_f \), are the \( t_pR_c \) values corresponding to the crossover points a, c and f, respectively. It is clear from Fig. 3 that the straight line is most cost-effective when \( v_a < t_pR_c < v_c \).

Finally, the binary hypercube is the most cost-effective topology when \( v_c < t_pR_c < v_f \). To find the value of \( v_a \), the CperTPU value for a straight line is set equal to the CperTPU value for a ring:

\[
\frac{mC_p + (m-1)C_L}{m[1-v_a(m+1)/3]} = \frac{mC_p + mC_L}{m(1-v_a m/2)}.
\]

Solving equation 17 for \( v_a \), it is found that

\[
v_a = \frac{-6C_L}{(m^2-2m)C_p + (m^2-5m)C_L}.
\]

Substituting \( m = 64 \) and \( C_p = C_L = 1 \) into equation 18, calculations reveal that \( v_a = -0.000775 \). The fact that \( v_a \) is negative means that the unidirectional ring is never the most cost-effective topology under the conditions of this example. By similar techniques, it is found that \( v_c = 0.0156 \) and \( v_f = 0.0965 \).

Example 2. Suppose that it is desired to determine the more cost-effective of the ring and the binary hypercube as \( R \), the 100% potential utilization message generation rate, is varied. It is assumed that the message generation rate is proportional to the potential utilization and that each network has 64 processors. It is further assumed that the cost of a processor, \( C_p \), and the cost of a link, \( C_L \), are both one arbitrary monetary unit. Finally, it is assumed that the processing times \( t_s \), \( t_t \) and \( t_a \) are 16, 32 and 16 time units, respectively.

Figure 4 shows a graph of CperTPU versus \( R \) for the conditions of this example. The crossover point has been labelled by the letter 'g', and the value of \( R \) corresponding to the crossover point is identified as \( v_g \). It is seen that the ring is more cost-effective when \( R < v_g \). For \( R > v_g \), however, the binary hypercube is more cost-effective.

To find the value of \( v_g \), the CperTPU value for a ring is set equal to the CperTPU value for the binary hypercube:

\[
\frac{mC_p + mC_L}{m\left(1 + v_g t_s + v_g t_t (m-2)/2 + v_g t_a\right)} = \frac{2^nC_p + n(2^{n-1})C_L}{2^n\left(1 + v_g t_s + v_g t_t (n-2)(2^{n-1}+1)/(2^n-1) + v_g t_a\right)}.
\]

Equation 19 was obtained by use of Table 2, and equations 10, 11, 13 and 15. Solving equation 19 for \( v_g \), it is found that

\[
v_g = \frac{1}{t_t \left(Q_1 Q_2 Q_3 - Q_2 Q_3 - Q_2 Q_4 \right) - t_s - t_a}.
\]

where

\[
Q_1 = mC_p + mC_L,
Q_2 = 2^nC_p + n(2^{n-1})C_L,
Q_3 = \frac{m-2}{2},
Q_4 = \left(n-2\right)2^{n-1} + 1.
\]

Substituting \( m = 64 \), \( n = 6 \), \( C_p = C_L = 1 \), \( t_s = t_t = t_a = 16 \) and \( t_t = 32 \) into equations 20 and 21, it is found that \( v_g = 0.0012063 \) messages per time unit.

The discussion in Examples 1 and 2 has assumed equal monetary costs and equal message relay times for nodes in different topologies. An avenue for further research would be to relax these assumptions in order to account for the reasonable feeling that the higher degree of a node in, say, a hypercube could add to the processor cost and/or the message relay times, relative to, say, a node in a ring. The higher message relay times for a hypercube might be due to the more complex nature of the routeing algorithm used as compared to that of a ring.
5. CONCLUSIONS

This paper represents a beginning in an effort to obtain some information for comparing the cost-effectiveness of multicomputers. Cost per unit total potential utilization, provides an inverse bang-per-buck measure that was used to compare the bidirectional straight line, unidirectional ring, square mesh and binary hypercube topologies. When messages are generated at a constant rate, the binary hypercube appears to be the most cost-effective at higher message rates, the square mesh at medium rates, and the straight line at lower rates. A perturbation analysis suggested that networks not adhering strictly to the alternate assumption that message generation rate be proportional to potential utilization may still behave in reasonable accordance with the theoretical model that underlies this paper.

Acknowledgment

The authors wish to express their appreciation to the referee who anonymously gave time to review the original manuscript of this paper. The constructive criticism and suggestions for improvement are highly appreciated and were also helpful for identification of issues for future research.

REFERENCES


APPENDIX A. Straight Line of m Processors with Message Rate Proportional to Potential Utilization

The solution can be expressed in matrix form as

\[ U = A^{-1}C, \]

where \( A \) is a non-singular matrix and \( U, A \) and \( C \) are matrices defined as follows:

\[
A = \begin{bmatrix}
    a_{11} & a_{12} & a_{13} & \ldots & a_{1m} \\
    a_{21} & a_{22} & a_{23} & \ldots & a_{2m} \\
    \vdots & \vdots & \vdots & \ddots & \vdots \\
    a_{m1} & a_{m2} & a_{m3} & \ldots & a_{mm}
\end{bmatrix},
\[
C = \begin{bmatrix}
    1 & 1 & 1 & \ldots & 1 \\
    1 & 1 & 1 & \ldots & 1 \\
    \vdots & \vdots & \vdots & \ddots & \vdots \\
    1 & 1 & 1 & \ldots & 1
\end{bmatrix},
\]

\[
U = \begin{bmatrix}
    U_1 \\
    U_2 \\
    \vdots \\
    U_n
\end{bmatrix}.
\]

The matrix \( C \) is a column matrix of \( m \) 1s. The individual components of matrix \( A \) are as follows:

\[
a_{ii} = 1 + R_{ii}, \quad \text{for} \quad 1 \leq i \leq m;
\]

\[
a_{ii} = R_{ii}[1/(m-1)], \quad \text{for} \quad 2 \leq i \leq m;
\]

\[
a_{mi} = R_{mi}[1/(m-1)], \quad \text{for} \quad 1 \leq i \leq m-1;
\]

\[
a_{ij} = R_{ij}[1/(m-1)] + R_{ji}[(m-i)/(m-1)], \quad \text{for} \quad 2 \leq i \leq m, 1 \leq j \leq i-1;
\]

\[
a_{ij} = R_{ij}[1/(m-1)] + R_{ji}[(i-j)/(m-1)], \quad \text{for} \quad 2 \leq i \leq m-1, i+1 \leq j \leq m.
\]

APPENDIX B. Square Mesh of \( m = n^2 \) Processors with Message Rate Proportional to Potential Utilization

The results that follow for a square mesh of \( m = n^2 \) processors are based upon what the authors call perimeter routing. Messages are always routed along the perimeter of the rectangle containing the origination \( (p_{orig}) \) and destination \( (p_{dest}) \) processors, as indicated in Fig. 5.

If \( \frac{1}{m} P_{i,j}, \) is the probability that the processor in row \( r \) and column \( c \) relays a message originating at the processor in row \( i \) and column \( j \), then

\[
\frac{1}{m} P_{i,j} = \sum_{k=1}^{m-1} C_k,
\]

where \( C_k \) is the number of contributions of magnitude \( 1/(m-1) \) to the probability due to consideration of case \( k \) in Fig. 5. Analysis of each case reveals the following expressions for these contributions:

\[
C_1 = (r \geq i) (c \leq j) (n-r) + (r \leq i) (c \geq j) (n-c) (n-r)
\]

\[
C_2 = (r \leq i) (c < j) (r-1) + (r \geq i) (c < j) (c-1) (r-1)
\]

\[
C_3 = (r \geq i) (c < j) (n-r) + (r \leq i) (c < j) (c-1) (n-r)
\]

\[
C_4 = (r \leq i) (c > j) (r-1) + (r \geq i) (c > j) (n-c) (r-1)
\]

\[
C_5 = (r \geq i) (c > j) (n-c)
\]

\[
C_6 = (r \leq i) (c < j) (c-1)
\]

\[
C_7 = (r \geq i) (c = j) (n-r)
\]

\[
C_8 = (r \leq i) (c = j) (r-1)
\]
Book Review

LYN ANTILL and CHRIS CLARE
Office Information Systems

Rapid improvements in technology and cost/performance ratios result in continuous changes in the way organisations apply information technology. This is true in office information systems as in other areas. While for most of the 1980s they were viewed largely as self-contained departmental systems, they are now, through the development of networks and integration with personal computing, corporate data processing and MIS, becoming part of or even the basis for organisational computing. Office systems seemed to be seeking a role throughout the later 80s; they may now have found it, as the platform for organisation-wide communications. This makes office systems more important than they have been before, and the need to understand them more pressing.

This book is intended as an introductory text for students in computing or business studies, or for people involved in implementing office systems either as managers or technical specialists. While it discusses current office technology including networks in some detail, it does not attempt to explain what office systems are for, or what the business benefits of OIS might be. This signals a general weakness on the business and organisational side.

On the technical side, the book contains a detailed discussion of word processing, image processing, electronic storage, electronic mail, networks, decision support, expert systems and system design. Computing students would find this accessible and useful, if the material is not covered in their other textbooks or courses. The discussion of office work however does not reflect its many-sidedness and complexity as in found in Doswell (1990), for example. It also does not provide them with the analysis of the function of the office and its role in the business which they particularly need. Instead, it focuses on office roles and activities as they are currently performed (and existing office equipment such as typewriters and filing cabinets), as the basis for the computer system; the implication therefore is that you automate existing procedures rather than analyse and redesign work processes and information flows, as is stressed for example in the recent Institute of Administrative Management/Touche Ross report on office automation. It would be unfortunate if a new textbook helped perpetuate among the next generation of computer specialists a view which is already becoming outdated.

The relation of office to other systems is now very topical, because of the trend to integration. On this the book is unclear. Office systems are treated as a category of MIS, along with transaction processing, and MIS are at one point equated to the total business information systems of the organisation. However, the rest of the discussion of MIS seems to use the term in the narrower and more usual sense of supporting management functions and decision-making requirements. Some attempt to classify systems, for example according to the routinelessness and structuredness of the work, would have been helpful.

Students of business and management and other non-specialists also need a clear understanding of the business role of office systems, as well as an explanation of the available technology. They would get something of the latter from this text; however, it is not always easy to follow. Technical terms are sometimes dropped in without explanation, some of the diagrams are difficult to interpret, and there is no glossary. The two case studies are rather limited, as one concerns the introduction of word processing in 1982, and the other desktop publishing in an academic environment.

The authors indicate the importance of the user at several points, but the book is neither business- nor user-led. It provides a better introduction to current office technology than to office information systems. The organisational aspects of OIS are mentioned in passing but not considered in any depth, and there is no vision of the important part which office systems will have in the near future as the basis of organisational communications.

FAITH NOBLE
Sheffield