Processor Equivalence for Daisy Chain Load Sharing Processors

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

A linear daisy chain of processors where processor load is divisible and shared among the processors is examined. It is shown that two or more processors can be collapsed into a single equivalent processor. This equivalence allows a characterization of the nature of the minimal time solution, a simple method to determine when to distribute load for linear daisy chain networks of processors without front end communication subprocessors and closed form expressions for the equivalent processing speed of infinitely large daisy chains of processors.

Manuscript received October 31, 1991; revised September 11, 1992.

IEEE Log No. T-AES/29/4/10988.

This research was supported by the National Science Foundation under Grant NCR-8703689 and by the SDIO/IST and managed by the U.S. Office of Naval Research under Grants N00014-85-K0610 and N00014-91-J4063.

Author's address: Dept. of Electrical Engineering, SUNY at Stony Brook, Stony Brook, NY 11794.

I. INTRODUCTION

Parallel computation has been of great interest in recent years. A parallel machine consists of a number of processors and an interconnection network to tie them together. This work examines a specific parallel processing problem on a specific architecture that allows the study of the integration of communication and computation. While these two issues are often studied separately, a combined study is rare.

The situation to be considered involves a linear daisy chain of processors, as is illustrated in Fig. 1. A single "problem" (or job) is solved on the network at one time. It takes time $w_i T_{cp}$ to solve the entire problem on processor i. Here w_i is inversely proportional to the speed of the ith processor and T_{cp} is the normalized solution time when $w_i = 1$. It takes time $z_i T_{cm}$ to transmit the entire problem representation (data) over the ith link. Here z_i is inversely proportional to the channel speed of the ith link and T_{cm} is the normalized transmission time when $z_i = 1$.

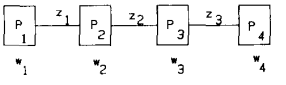

Fig. 1. Linear chain of processors.

It is assumed that the problem representation can be divided amongst the processors. Thus the problem representation is said to be "divisible". That is, fraction α_i of the total problem is assigned to the ith processor so that its computing time becomes $\alpha_i w_i T_{cp}$. It is desired to determine the optimal values of the α_is so that the problem is solved in the minimum amount of time. The situation is nontrivial as there are communication delays incurred in transmitting fractional parts of the problem representation to each processor from the originating processor.

There is a good deal of literature on scheduling and load sharing in multiprocessors [5–7, 14–21]. However most work to date assumes that a job can be assigned to at most one processor. Only recently has there been interest in multiprocessor scheduling with jobs that need to be assigned to more than one processor [22–24]. In this work, one has a single job that can be arbitrarily partitioned among a number of processors. The framework being described is particularly germane to processing involving large data files (so that communication delay is nonnegligible), such as sensor data processing, signal processing, image processing, and Kalman filtering, where the data can be divided among multiple processors.

Two cases are considered: processors that have front end communications subprocessors for communications off-loading so that communication and computation may proceed simultaneously,

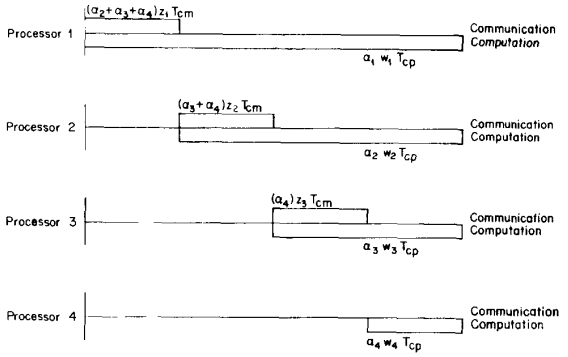

Fig. 2. Timing diagram: Network with front-end communications subprocessors.

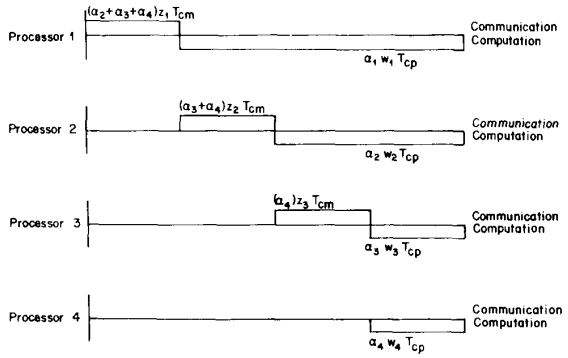

Fig. 3. Timing diagram: Network without front-end communications subprocessors.

and processors without front end communications subprocessors so that communication and computation must be performed at separate times.

A timing diagram for a linear daisy chain network of four processors with front-end communications subprocessor (as in Fig. 1) is illustrated in Fig. 2. There is one graph for each processor. The horizontal axis is time. The upper half of each graph indicates communication time and the lower half indicates computation time. It is assumed that the problem (load) originates at the left most processor.

At time 0, processor 1 can start working on its fraction, α_1, of the problem in time $\alpha_1 w_1 T_{cp}$. It also simultaneously communicates the remaining fraction of the problem to processor 2 in time $(\alpha_2 + \alpha_3 + \alpha_4) z_1 T_{cm}$. Processor 2 can then begin computation on its fraction of the problem (in time $\alpha_2 w_2 T_{cp}$) and communicates the remaining load to processor 3 in time $(\alpha_3 + \alpha_4) z_2 T_{cm}$. The process continues until all processors are working on the problem.

A similar, but not identical, situation for a linear daisy chain network with processors that do not have front-end communication subprocessors is illustrated in Fig. 3. Here each processor must communicate the remaining load to its right neighbor before it can begin computation on its own fraction.

In [1] recursive expressions for calculating the optimal α_is 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. Intuitively this is because otherwise some processors would be idle while others were still busy. Analogous solutions have been developed for tree networks [2] and bus networks [3, 4]. 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 [10]. Optimal sequences of load distribution in tree networks are described in [8, 9, 11]. Closed form solutions for homogeneous bus and tree networks appear in [13].

The concept of collapsing two or more processors and associated links into a single processor with equivalent processing speed is presented here. This allows a complete proof (an abridged one appears in [1]) that for the optimal, minimal time solution all processors must stop at the same time. Moreover, for the case without front end communications subprocessors, it allows a simple algorithm, described in Section III, to determine when it is economical to distribute load amongst multiple processors. Finally, in Section IV, the notion of equivalent processors enables the derivation of simple closed-form expressions for the equivalent speed of a linear daisy chain network containing an infinite number of processors. This provides a limiting value for the performance of this network architecture and load distribution sequence [11].

II. EQUIVALENT PROCESSORS

Consider a linear daisy chain network of N processors as in Fig. 1. 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 the original two processors. Two cases, processors with and without front-end communication subprocessors, are considered.

In both cases it is assumed that the load originates at the left-most processor (processor 1). If the load originates at an interior processor one can use the same methodology to collapse the processors to the left and the right of the originating processor into equivalent processors and then collapse the remaining three processors into a single equivalent processor.

A. Front End Communications Subprocessors

We start with the $N-1$st and Nth processors, as illustrated in Fig. 4. The figure begins at the moment when the load has finished being transmitted to the $N-1$st processor from the $N-2$nd processor. As in [1], the $N-1$st processor keeps $\hat{\alpha}_{N-1}$ fraction of what it receives and transmits the remaining $1-\hat{\alpha}_{N-1}$

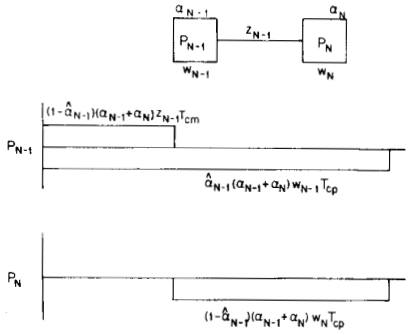

Fig. 4. Timing diagram for $N-1$st and Nth processors with front-end communications subprocessors.

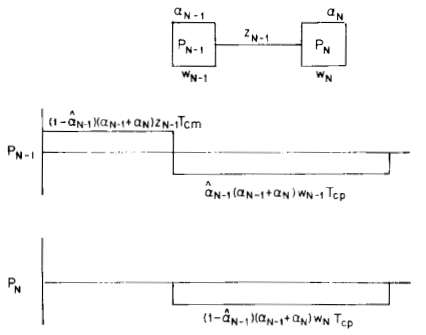

Fig. 5. Timing diagram for $N-1$st and Nth processors without front-end communications subprocessors.

fraction to the Nth processor. The total load received by the $N-1$st processor from the $N-2$nd is $(\alpha_{N-1}+\alpha_N)$. The time each is active, from the figure, is

$$T_{N-1} = \hat{\alpha}_{N-1}(\alpha_{N-1}+\alpha_N)w_{N-1}T_{cp} \tag{1}$$

$$T_N = (1-\hat{\alpha}_{N-1})(\alpha_{N-1}+\alpha_N)z_{N-1}T_{cm} + (1-\hat{\alpha}_{N-1})(\alpha_{N-1}+\alpha_N)w_N T_{cp}. \tag{2}$$

To prove that the minimal time solution occurs when both processors stop at the same time, the possibilities $T_{N-1} \geq T_N$ and $T_{N-1} \leq T_N$ must be examined. If $T_{N-1} \geq T_N$, simple algebra results in

$$\hat{\alpha}_{N-1} \geq \frac{z_{N-1}T_{cm} + w_N T_{cp}}{w_{N-1}T_{cp} + z_{N-1}T_{cm} + w_N T_{cp}} \tag{3}$$

with equality occurring when both processors stop at the same time. Minimizing the solution time, $T_{\text{sol}} = T_{N-1}$, clearly requires

$$\min T_{\text{sol}} = (\min \hat{\alpha}_{N-1})(\alpha_{N-1}+\alpha_N)w_{N-1}T_{cp} \tag{4}$$

so that the optimal value of $\hat{\alpha}_{N-1}$ occurs for equality in (3). The quantity $(\alpha_{N-1}+\alpha_N)$ is not involved in the minimization since the value of $\hat{\alpha}_{N-1}$ is unaffected by the total load, $(\alpha_{N-1}+\alpha_N)$, delivered to the $N-1$st processor. Put another way, the optimization involves the fraction of load being allocated between P_{N-1} and P_N, not the total load allocated to these two processors. The other half of the proof, for $T_{N-1} \leq T_N$, is similar.

The two processors with front-end (fe) communications subprocessors can be replaced by a single processor with equivalent speed constant:

$$w_{\text{eq}}^{\text{fe}} = \frac{\hat{\alpha}_{N-1}(\alpha_{N-1}+\alpha_N)w_{N-1}T_{cp}}{(\alpha_{N-1}+\alpha_N)T_{cp}} = \hat{\alpha}_{N-1}w_{N-1}. \tag{5}$$

Here $\hat{\alpha}_{N-1}$ is given by (3) with equality. The solution time is divided by the normalized computation time to yield the equivalent speed constant. Thus, starting with the $N-1$st and Nth processors, the entire linear chain of processors can be collapsed, two at a time, into a single equivalent processor.

Thus one can recursively show that for a network of N processors the optimal solution occurs when all processors stop at the same time.

B. No Front-End Communications Subprocessor

Again, consider the $N-1$st and Nth processors in a linear chain. Fig. 5 starts from the moment when load has finished being transmitted from the $N-2$nd to the $N-1$st processor. Again, as in [1], 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 Nth processor. From Fig. 5, the time each is active is

$$T_{N-1} = (1-\hat{\alpha}_{N-1})(\alpha_{N-1}+\alpha_N)z_{N-1}T_{cm} + \hat{\alpha}_{N-1}(\alpha_{N-1}+\alpha_N)w_{N-1}T_{cp} \tag{6}$$

$$T_N = (1-\hat{\alpha}_{N-1})(\alpha_{N-1}+\alpha_N)z_{N-1}T_{cm} + (1-\hat{\alpha}_{N-1})(\alpha_{N-1}+\alpha_N)w_N T_{cp} \tag{7}$$

Once again, to prove that the minimal time solution requires both processors to stop at the same time, the cases $T_{N-1} \geq T_N$ and $T_{N-1} \leq T_N$ can be considered. For $T_{N-1} \geq T_N$, simple algebra results in

$$\hat{\alpha}_{N-1} \geq \frac{w_N}{w_{N-1}+w_N} \tag{8}$$

with equality occurring when both processors stop at the same time. From (6) the solution time can be rewritten as

$$T_{\text{sol}} = T_{N-1} = (\alpha_{N-1}+\alpha_N)z_{N-1}T_{cm} + \hat{\alpha}_{N-1}(\alpha_{N-1}+\alpha_N)(w_{N-1}T_{cp} - z_{N-1}T_{cm}). \tag{9}$$

The sign of the term $(w_{N-1}T_{cp} - z_{N-1}T_{cm})$ now becomes important. If it is positive, minimizing T_{sol} is equivalent to minimizing $\hat{\alpha}_{N-1}$ and the optimal solution occurs at equality for (8). In other words, if $w_{N-1}T_{cp} > z_{N-1}T_{cm}$, communication is fast enough relative to computation that the distribution of load is economical. Again, $(\alpha_{N-1}+\alpha_N)$ is not involved in the minimization.

On the other hand, if $(w_{N-1}T_{cp} - z_{N-1}T_{cm})$ is negative, then minimizing T_{sol} is equivalent to maximizing $\hat{\alpha}_{N-1}$ at $\hat{\alpha}_{N-1} = 1$. That is, communication speeds are slow relative to computation speed so that it is more economical for processor $N-1$ to process the entire load itself rather than to distribute part of it to processor N.

The case where $T_{N-1} \leq T_N$ proceeds along similar lines. Again, the ability to collapse processors into equivalent processors allows one to extend the proof that two processors must stop at the same time for a minimal time solution to N processors.

III. WHEN TO DISTRIBUTE LOAD

A practical problem for the case without front-end communications subprocessor is to compute the equivalent computation speed of a linear daisy chain network when, in fact, the optimal solution may not make use of all processors, because of too slow communication speeds. Again, if the load originates at the left-most processor, this can be done by collapsing the processors, two at a time, from right to left in Fig. 1, into a single equivalent processor. However, when looking at two adjacent processors, say the $i-1$st and the ith (where the ith is an equivalent processor for processors $i, i+1, \ldots$), one must determine whether or not it is economical to distribute load. That is, one seeks the faster of either the solution with both processors, T_{both}, or with just the single $i-1$st processor, T_{single}:

$$T_{both} = (1-\hat{\alpha}_{i-1})(\alpha_{i-1}+\alpha_i)z_{i-1}T_{cm} + \hat{\alpha}_{i-1}(\alpha_{i-1}+\alpha_i)w_{i-1}T_{cp} \quad (10)$$

$$T_{single} = (\alpha_{i-1}+\alpha_i)w_{i-1}T_{cp}. \quad (11)$$

Here fraction $\hat{\alpha}_{i-1}$ of the total load, $(\alpha_{i-1}+\alpha_i)$, is assigned to processor $i-1$ and fraction $1-\hat{\alpha}_i$ is assigned to processor i. If $T_{single} < T_{both}$ then the ith processor is removed from consideration and the equivalent processing speed constant, with no front end (nfe) communication subprocessor, is

$$w_{eq}^{nfe} = \frac{(\alpha_{i-1}+\alpha_i)w_{i-1}T_{cp}}{(\alpha_{i-1}+\alpha_i)T_{cp}} = w_{i-1}. \quad (12)$$

If $T_{single} > T_{both}$ then load distribution is economical and the two processors are collapsed into a single equivalent processor with speed constant:

$$w_{eq}^{nfe} = \frac{(1-\hat{\alpha}_{i-1})z_{i-1}T_{cm} + \hat{\alpha}_{i-1}w_{i-1}T_{cp}}{T_{cp}}. \quad (13)$$

From (8):

$$\hat{\alpha}_{i-1} = \frac{w_i}{w_{i-1}+w_i}. \quad (14)$$

Note that in (13) factors of $(\alpha_{i-1}+\alpha_i)$ cancel in the numerator and denominator.

By keeping track of which of (10) and (11) is smaller, it is possible to determine which processors to remove from the final network.

Note that the above procedure can also be applied to the situation when the load originates at a processor which is located in the interior of the network. The parts of the network to the left and to the right of the originating processor can be collapsed, into equivalent processors, following the previous procedure. The remaining three processors (left, originating, right) can then be further collapsed into a single equivalent processor. Naturally, it must be checked whether the inclusion of the left and/or right equivalent processor leads to a faster solution.

IV. INFINITE NUMBER OF PROCESSORS

A difficulty with the linear network daisy chained architecture is that as more and more processors are added to the network, the amount of improvement in the equivalent speed of the network approaches a saturation limit. Intuitively, this is because of the overhead in communicating the problem representation down the linear daisy chain in what is essentially a store and forward mode of operation.

It is possible to develop simple expressions for the equivalent processing speed of an infinite number of homogeneous processors and links. These provide a limiting value on the performance of this architecture. The technique is similar to that used for infinitely sized electrical networks to determine equivalent impedance.

Let the load originate at a processor at the left boundary of the network (processor 1). The basic idea is to write an expression for the speed of the single equivalent processor for processors $1,2\ldots\infty$. This is a function of the speed of the single equivalent processor for processors $2,3\ldots\infty$. However these two speeds should be equal since both involve an infinite number of processors. One can simply solve for this speed.

Consider, first, the case where each processor has a front-end communication sub-processor. Let $w_i = w$ and $z_i = z$. Let the network consist of P_1 and an equivalent processor for processors $2,3\ldots\infty$. Then:

$$w_{eq}^{fe} = \hat{\alpha}_1 w. \quad (15)$$

But from (3) with equality, and making the above assumption,

$$w_{eq}^{fe} = \frac{z\rho + w_{eq}^{fe}}{w + z\rho + w_{eq}^{fe}}w \quad (16)$$

where $\rho = T_{cm}/T_{cp}$. Solving for w_{eq}^{fe} results in

$$w_{eq}^{fe} = \frac{-z\rho + \sqrt{(z\rho)^2 + 4wz\rho}}{2} \quad (17)$$

The solution time for such an infinite network is simply given by $T_{sol} = w_{eq}^{fe}T_{cp}$.

In a similar manner, an expression for the equivalent processing speed of a linear daisy chain

network with an infinite number of processors with no front-end communication subprocessors can be determined. Again, the load originates at processor 1 at the left boundary of the daisy chain.

$$w_{eq}^{nfe} = \sqrt{wz\rho}. \tag{18}$$

The solution time for this infinite network is simply given by $T_{sol} = w_{eq}^{nfe} T_{cp}$.

This last expression is somewhat intuitive. Doubling w and z doubles w_{eq}^{nfe}. Doubling either w or z alone increases w_{eq}^{nfe} by a factor of $\sqrt{2}$. These results agree very closely with numerical results presented in [1]. It is straightforward to show that $w_{eq}^{fe} < w_{eq}^{nfe}$. Thus, in this limiting case, solution time is always reduced through the use of front-end processors.

It is also possible to use the above results to calculate the limiting performance of an infinite sized daisy chain when the load originates at a processor at the interior of the network (with the network having infinite extent to the left and the right). Expressions (17) or (18) can be used to construct equivalent processors for the parts of the network to the left and right of the originating processor. The resulting three processor system can then be simply solved [1, 12].

V. CONCLUSION

The concept of collapsing two or more processors into an equivalent processor has been shown to be useful in examining a variety of aspects related to these linear daisy chain networks of load sharing processors. Expressions for the performance of infinite chains of processors are particularly useful as if one can construct a finite-sized daisy chain that approaches the performance of a hypothetical infinite system, one can feel comfortable that performance cannot be improved further for this particular architecture and load distribution sequence [11].

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 a 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 network with communication delays.
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 & Cybernetics; Special Issue on Distributed Sensor Nets, **21** (Sept. 1991).

[5] Leung, J. Y.-T., and Young, G. H. (1989)
Minimizing schedule length subject to minimum flow time.
SIAM Journal on Computing, **18** (Apr. 1989), 314–326.

[6] Coffman, Jr., E. G., Gavey, M. R., and Johnson, D. S. (1978)
An application of bin packing to multiprocessor scheduling.
SIAM Journal on Computing, **7** (Feb. 1978).

[7] Coffman, Jr., E. G., and Sethi, R. (1976)
Algorithms minimizing mean flow time: Schedule length properties.
Acta Informatica, **6** (1976), 1–14.

[8] Kim, H. J., Jee, G.-I., and Lee, J. G.
Optimal load distribution for tree network processors.
To be published.

[9] Bharadwaj, V., Ghose, D., and Mani, V.
Closed-form solutions for optimal processing time in distributed single-level tree networks with communication delays.
To be published.

[10] Mani, V., and Ghose, D.
Distributed computation in a linear network: Closed form solutions and computational techniques.
To be published.

[11] Bharadwaj, V., Ghose, D., and Mani, V.
A new strategy of load distribution in a distributed single-level tree network with communication delays.
To be published.

[12] Bataineh, S., and Robertazzi, T. G. (1992)
Ultimate performance limits 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.

[13] Bataineh, S., and Robertazzi, T. G. (1992)
Closed form solutions for bus and tree networks of processors load sharing a divisible job.
Technical Report 627, SUNY at Stony Brook College of Engineering and Applied Science, May 27, 1992. Available from the author.

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

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

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

[17] Ramamritham, K., Stankovic, J. A., and Zhao, W. (1989)
Distributed scheduling of tasks with deadlines and resource requirements.
IEEE Transactions on Computers, **C-38** (Aug. 1989), 1110–1122.

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

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

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

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

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

[23] 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.

[24] 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.

Thomas G. Robertazzi (S'75—M'81—SM'91) received the B.E.E. degree from the Cooper Union, New York, NY, in 1977 and the Ph.D. degree in electrical engineering from Princeton University, Princeton, NJ, in 1981.

During 1982–1983 he was an Assistant Professor of Electrical Engineering at Manhattan College, Riverdale, NY. Since 1983 he has taught and conducted research at the Electrical Engineering Department of the State University of New York at Stony Brook, where he is presently an Associate Professor. During the Fall of 1990, Prof. Robertazzi was a Visiting Research Scientist at Columbia University's Center for Telecommunications Research.

His research interests are in the area of the performance evaluation of communication and computer systems. Dr. Robertazzi has been a member of the INFOCOM Technical Program Committee since 1989.